Automatización y Control de Forma Remota

del Observatorio Astronómico de la Universidad Tecnológica de Pereira

Jefferson Martínez Orozco

Daniel Felipe Henao Toro

Universidad Tecnológica de Pereira

Facultad de Ingenierías: Eléctrica, Electrónica, Física y Ciencias de la Computación

Programa de Ingeniería Electrónica

Pereira, Risaralda - Colombia

5 de Noviembre de 2013

Automatización y Control de Forma Remota

del Observatorio Astronómico de la Universidad Tecnológica de Pereira

Proyecto de grado para optar al título de Ingenieros Electrónicos

Director

MSc. Edwin Andrés Quintero Salazar

Universidad Tecnológica de Pereira

Facultad de Ingenierías: Eléctrica, Electrónica, Física y Ciencias de la Computación

Programa de Ingeniería Electrónica

Pereira, Risaralda - Colombia

5 de Noviembre de 2013

Nota de aceptación

_____
_____
_____
_____

_____

Firma del Director

Pereira 5 de Noviembre de 2013

_____

Firma del Jurado

Dedicado a Dios, a nuestros padres, hermanos, familiares y amigos.

Agradecimientos

Este trabajo no habría sido posible sin la influencia directa o indirecta de muchas personas a las que agradezco profundamente por estar presentes en las distintas etapas de su elaboración, así como en el resto de mi vida.

Agradecemos al profesor Edwin Andrés Quintero por dirigir nuestro trabajo de grado, por su confianza, colaboración y apoyo en el proceso de realización de este proyecto.

A todos los docentes de la Universidad Tecnológica de Pereira que compartieron sus conocimientos, dentro y fuera de clase, haciendo posible que nuestra formación profesional se resuma en satisfacciones académicas e inquietudes insatisfechas en continua indagación.

A nuestros amigos y compañeros. A quienes trabajaron con nosotros hombro a hombro durante varios años, poniendo lo mejor de su energía y empeño por el bien de nuestra formación profesional, a quienes compartieron su confianza, tiempo, y los mejores momentos que vivimos durante esta etapa como estudiantes de pregrado, dentro y fuera del campus.

Por último a nuestras familias y seres más queridos, en especial a nuestros padres por brindarnos su comprensión y apoyo incondicional durante toda nuestra carrera, por sus consejos que nos orientaron a tomar las mejores decisiones y sobre todo por creer en nosotros.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Anexos

Resumen

El desarrollo de aplicaciones web ha tenido un gran crecimiento a lo largo de la última década, llevando un sin número de servicios a los diferentes usuarios que en todo el mundo tienen acceso a esta red, este crecimiento no ha sido ajeno al estudio de la astronomía, logrando desarrollar incluso importantes aplicativos web para el estudio interactivo y simulado de los astros descubiertos hasta la fecha.

Gracias a esta evolución y a la creación y mejoramiento de herramientas para el desarrollo web, se puede lograr construir un importante sistema de control vía web para el observatorio astronómico de la Universidad Tecnológica de Pereira.

Este sistema, que básicamente está compuesto de dos componentes que se desarrollaron de forma paralela; un componente de software que contiene un aplicativo web donde los diferentes usuarios con los permisos y privilegios adecuados pueden ejercer de forma segura, un adecuado control vía web sobre los dispositivos que se encuentran en el observatorio, dando control a la apertura y cierre de las compuertas, al giro del domo, al encendido de una luminaria y por supuesto al movimiento del telescopio, sin dejar atrás la obtención de video en streaming del observatorio, el uso de un servidor de correo que permite enviar la información de forma automática desde la web y al acceso a la información entregada por una estación meteorológica instalada en el observatorio.

El segundo componente está integrado por todos los dispositivos de hardware utilizados para el logro de este objetivo, incluyendo los sensores adecuados para cada necesidad en particular, los sistemas o módulos de comunicación inalámbrica como los XBee y los módulos de radio frecuencia, además del uso de un microcontrolador como cerebro fundamental para la interacción entre los diferentes componentes del sistema, sin dejar de lado el desarrollo de la etapa de potencia y control, con el uso de diferentes dispositivos electrónicos.

Finalmente se pretende mediante el uso de tecnología de fácil acceso, lograr un control vía web suficientemente adecuado sobre el observatorio astronómico de la universidad Tecnológica de Pereira, logrando así que diferentes estudiantes, profesores e investigadores, puedan tener acceso a esta importante herramienta desde lugares remotos, potencializando así el estudio de los diferentes fenómenos astronómicos.

Introducción

Actualmente el desarrollo de software es una herramienta esencial que proporciona diversas soluciones innovadoras en cualquier campo de acción. Según lo anterior se vio la necesidad de implementar un software a la medida, el cual permitiera dar solución al control sobre el observatorio astronómico por medio de un aplicativo web.

El aplicativo web es una plataforma donde se administra el acceso al control de los dispositivos y aplicaciones del observatorio Astronómico de la Universidad Tecnológica de Pereira, donde los usuarios pueden acceder en diferentes horarios, que previamente han sido aprobados por el administrador de la plataforma.

El aplicativo o plataforma web permite al administrador crear permisos y privilegios, mantener la conexión con el hardware y proveer por medio de un protocolo diversas acciones como; el control a la apertura y cierre de las compuertas, al giro del domo y el encendido de una luminaria, además de proveer una imagen en línea de las aplicaciones que permiten el control del telescopio, la lectura de información climática y la toma de fotografía estelar, sin dejar de lado la obtención de video en streaming del observatorio.

Además de proveer dichas funcionalidades, el aplicativo permite informar los sucesos y actividades del administrador y de los usuarios, como las notificaciones o respuestas a las peticiones que el usuario realiza sobre el administrador, por ejemplo, informar si una solicitud de registro o acceso es aprobada, para ello se usa un servidor de correo electrónico que permite enviar la información de forma automática desde el aplicativo web al usuario.

Para lograr el sistema de control vía web del Observatorio Astronómico, también se hizo necesario el desarrollo de una plataforma de hardware, que permitiera interactuar con los dispositivos de control manual con los que ya contaba el Observatorio, como por ejemplo el motor del domo y los motores de las compuertas. Para esto se utilizaron diferentes módulos que ejecutan tares de transmisión de datos y acciones de control sobre los dispositivos.

El hardware se controla por medio de un sistema central basado en un microcontrolador Atmel montado sobre un módulo Arduino. Este microcontrolador procesa la información recibida tanto de los diferentes sensores como del usuario que interactúa con el aplicativo web, de modo que, de acuerdo a la lógica programada en dicho microcontrolador, se toman las decisiones de control sobre los diferentes módulos o actuadores conectados a los dispositivos finales.

1

Preliminares

1.1. Planteamiento del Problema

La utilización de los observatorios astronómicos para el desarrollo de actividades de aprendizaje e investigación en el contexto universitario, son hoy en día una herramienta de alta importancia para los profesores, estudiantes de pregrado y postgrado de las diferentes áreas de ingeniería de la universidad.

Como es bien sabido, la Universidad Tecnológica de Pereira (UTP) cuenta con un observatorio astronómico para el desarrollo de dichas actividades, sin embargo, es importante reconocer la limitación que existe con respecto al acceso a esta importante herramienta, debido a las restricciones de horarios en los que este puede ser operado.

En la actualidad, el observatorio de la UTP no cuenta con un sistema automatizado ni informatizado, que permita tener control de manera remota del sistema, restringiendo su uso únicamente a una forma presencial y a unos horarios específicos, que en muchas ocasiones no son suficientes para atender la demanda de profesores, estudiantes o la realización de actividades orientadas a difundir la astronomía o a contribuir a la comunidad científica.

Estas limitaciones de acceso al observatorio astronómico, no permiten, para las exigencias del mundo actual, un desarrollo adecuado de las investigaciones de carácter científico, ni de la apropiación efectiva de conocimientos relacionados con la astronomía, y minimiza la posibilidad de potencializar el crecimiento tecnológico y científico de la región, con la ayuda de las herramientas tecnológicas de la universidad.

Además se debe considerar que la disponibilidad de prácticas que necesitan profesionales idóneos de pregrado y postgrado en astronomía en Colombia es baja, y está sujeta a ciertas restricciones como las relacionadas con desplazamiento de una ciudad a otra o la existencia de condiciones climáticas adversas.

1.2. Justificación

Actualmente los observatorios astronómicos existentes en las Universidades en todo el mundo, tienen diversas opciones de acceso al público. La Universidad Tecnológica de Pereira no ha sido ajena a esta situación y en la actualidad presenta un sistema de acceso que requiere la presencia del personal interesado.

Se hace necesario entonces la elaboración de un sistema automatizado y el desarrollo de una plataforma informática, mediante el uso de herramientas tecnológicas acordes a las necesidades y a los recursos existentes en la Universidad. Así, el diseño y la ejecución de este sistema, permitirá el acceso remoto al observatorio astronómico, ofreciendo diversas actividades que difundan la investigación astronómica y el desarrollo de labores académicas en diferentes ambientes y espacios relacionados con el estudio de la astronomía.

La implementación de la plataforma informática y de control, admite además el buen aprovechamiento del tiempo en el que el observatorio astronómico puede ser operado, sin limitar su uso a horarios específicos y evita la dependencia en cuanto a la presencia de la persona encargada de la supervisión.

Por lo tanto, el desarrollo de un observatorio astronómico automatizado, con su respectiva interfaz de control vía web, otorgaría un grado de reconocimiento a la universidad tanto a nivel regional como nacional, por la aplicación de avances tecnológicos en favor de la investigación, la ciencia y la comunidad en general.

De esta manera se asumen retos a futuro claros, como consolidar al observatorio astronómico como un centro de investigación con prestigio nacional, reforzando además, la presencia social en la comunicación pública de la astronomía.

1.3. Objetivos

Los objetivos de este trabajo se conforman de un solo objetivo general, que se logra una vez alcanzados los tres objetivos específicos que se exponen a continuación.

1.3.1. Objetivo General

Implementar un sistema de control vía web del observatorio astronómico de la Universidad Tecnológica de Pereira.

1.3.2. Objetivos Específicos

1. Diseñar y construir un circuito que sirva como interfaz de comunicación entre un ordenador local y los motores de la cúpula del observatorio.

2. Elaborar un software apropiado con el que se logre operar el circuito de control de motores de la cúpula, desde un computador.

3. Implementar una solución, que permita utilizar el software existente de control de movimiento del telescopio (AutostarWeb), desde un sitio remoto a través de un computador.

4. Diseñar y construir una plataforma Web que integre los parámetros de control de los motores de la cúpula del observatorio, el contenido de los indicadores de la estación meteorológica y la información visual obtenida desde el software de control del movimiento del telescopio (AutostarWeb).

1.4. Estado del Arte

Son varios los trabajos que se han realizado en el tema de control vía web de hardware remoto y más específicamente en observatorios astronómicos, entre ellos podemos encontrar los siguientes:

En 1995, los ingenieros Kenzi Watanabe e Hiroshi Okada miembros de las Universidades de Saga y Wakayama en Japón respectivamente, plantearon la primera propuesta de desarrollo de un control vía Web del telescopio del observatorio Misato de Japón [1]. La necesidad partió del interés que tenían los estudiantes de escuelas sobre objetos y fenómenos astronómicos; y a su vez, de la dificultad de enseñar astronomía en los centros escolares, ya que sus estudiantes no podían observar objetos astronómicos en tiempo real. Lograron entonces, la implementación de su propuesta y la puesta a prueba, en la que se consiguió observar el video por Internet en tiempo real de un inusual fenómeno (la desaparición del anillo de Saturno), ya que el próximo fenómeno visible según los expertos, sucedería aproximadamente a los 45 años; tres veces el periodo de tiempo anterior, caso en que se había demorado casi 15 años. Lo anterior permitió mostrar que los beneficios de acceso a través de Internet y de algunas aplicaciones multimedia, son herramientas con grandes ventajas para dictar las lecciones de astronomía en las aulas de clase, ya que se podrían observar simultáneamente en cada computador enlazado a Internet los cambios de las transiciones de algunos fenómenos astronómicos tales como eclipses lunares (transmitidas desde otros países ya que el horario de escuelas no coincide con la hora de ocurrencia de estos fenómenos) y eclipses solares, o bien estrellas que solo se pueden observar en distantes zonas geográficas diferentes a las de Japón.

La NASA, en 1997 a través del ingeniero Adam Rifkin planteó el diseño de un sistema de control para el telescopio Hubble por medio de Internet [2], por considerarse este el más sofisticado, ya que su ubicación en el espacio es idealmente correcta, puesto que la contaminación lumínica es mínima. Desde entonces el proyecto lo inició la GFSC (La sala de misión de operaciones de ingenieros especializados), rediseñando su funcionalidad e incluyendo el sistema de control vía Internet, ya que por medio de este, la transferencia de información seria eficiente y lograría ser más comprensible para el usuario gracias a su interfaz gráfica. Uno de los objetivos principales por el cual el proyecto era viable, es la reducción de costos mediante la automatización de las operaciones de rutina a través de la tecnología de sistemas expertos, ya que los ingenieros de las misiones espaciales usualmente se requieren para intervenir sobre alguna anomalía en la nave espacial, o para brindar atención de situaciones que se planteen según se determine en la estación base de tierra.

Este nuevo sistema está ahora operando y corriendo en paralelo con el existente en tierra, junto a la instalación de una cámara que permite aumentar la visión del Hubble, este sistema fue actualizado en el 2002 por el mismo diseño de reingeniería de control vía web, cuya finalización se estimaba para 3 años después, es decir, el 2005.

En el año 2009, los Ingenieros DONG Jian, JIN Ge, DENG Xiao-chao y YUAN Hai-long del departamento de física moderna de la Universidad de ciencia y Tecnología de Hefei, China; presentaron el diseño de un software (Interfaz de comunicación de usuario remoto con el software del Observatorio), con una estructura más compleja por medio de grandes capas de control como la aplicación de usuario y el agente de aplicación (Interfaz entre usuario y el subsistema que había allí en ese observatorio) [3]; el cual proveía más opciones de manejo para un Sistema de control de Observatorio de mayor escala, reutilizando las aplicaciones de uno de menor escala, ya que se requería la ampliación del observatorio con un telescopio de mayor cobertura.

En Diciembre del 2009, los Ingenieros Diego López, Raquel Cedazo, Francisco Manuel Sánchez y José María Sebastián de la Universidad Pontificia de Comillas y Universidad Politécnica de Madrid respectivamente, diseñaron un sistema de control especializado (Ciclope), el cual se comunica con un usuario remoto por medio de un *Browser Web* y permite el manejo de un hardware especializado para la ejecución de tareas específicas (robot) [4]. Este diseño Web es una guía para los estudiantes que permite una mejor comprensión de teorías más abstractas, desarrollando sus habilidades prácticas y mejorando su capacidad para analizar y resolver problemas. Este proyecto en conjunto se construyó con dos características, la primera es el enfoque de colaboración, que permite a los estudiantes colaborarse mientras hacen sus trabajos de laboratorio con herramientas como el chat, foros, entre otras; y la segunda es el uso de una fuente y una metodología de contenido abierto, cuyo principal beneficio es que cualquier persona pueda reproducir, mejorar o integrar el mismo laboratorio web en un entorno de aprendizaje más cómodo. Esta aplicación sigue siendo viable y se introdujo en la astronomía ya que es una ciencia compleja, por lo que para trabajar en ella se necesita de la visualización de fenómenos para lograr una mejor comprensión; incluso, cuando uno de los principales problemas, corresponden a la falta de recursos disponibles sobre el lugar de observación; o bien, la escasez de personal o material que permitan el ingreso a este lugar.

En el año 2010, el observatorio de Montegancedo junto con los mismos ingenieros del proyecto Ciclope propusieron el diseño de un sistema de control vía Web, completamente interactivo y libre para el acceso al público del observatorio, a través de una aplicación Web 2.0 [5], con el fin de lograr que los usuarios remotos entrenen con dispositivos astronómicos reales, permitiendo abrir los conocimientos de esta ciencia a la sociedad. Una de los principales avances de este sistema, fue el aumento en la colaboración y la participación a través de los actuales servicios

conocidos en Internet, logrando importantes avances en diferentes aspectos, como la gran cantidad de usuarios registrados, además de los contenidos cargados por la participación de buena calidad de los usuarios, tales como escritos, mensajes, votos y etiquetas.

De manera independiente, los Ingenieros Japoneses Kenzy Watanabi, Hisaharu Tanaka, Makoto Otani y Misami Okyudo de las Universidades de Saga y Wakayama de Japón respectivamente, plantearon la misma propuesta que los ingenieros de la universidad de Montegancedo, con una diferencia; estos han desarrollado un simple y compacto sistema de control remoto de dos telescopios ubicados, uno en U.S.A. y otro en Japón [6], permitiendo a los estudiantes de centros de investigación en Japón realizar las prácticas en el día y en la noche, ya que existe la disponibilidad de dos telescopios en dos extremos del globo terráqueo con diferente zona horaria. En el diseño de esta implementación basada en un control Web, caracterizado por un entorno sencillo de costo reducido, se utilizó un telescopio ecuatorial comercial de marca *Vixen*, una cámara compacta CCD de video y una pequeña interfaz de Linux para el sistema de telescopio mandado a distancia.

Los anteriores desarrollos se plantearon y ejecutaron debido a la demanda existente de personas interesadas en el estudio y la observación astronómica, ya que esta viene creciendo de forma continua en todo el mundo, a pesar de la complejidad en la realización de las prácticas y la baja disponibilidad de equipos. Se ha dado entonces un salto importante para que a través del Internet se logre avanzar en este conocimiento de una manera más ágil, logrando que tanto los conocedores del tema como los usuarios comunes, tengan con facilidad el acceso a herramientas de la astronomía que antes no estaban a su alcance, y a su vez adquirir mayor entendimiento gracias a este método interactivo de aprendizaje.

2

Levantamiento de Planta

Para realizar un control de acceso remoto adecuado para el observatorio ubicado en la universidad tecnológica de Pereira, es necesario describir a cabalidad los elementos de software y hardware con los cuales se cuenta para realizar la implementación, estos elementos se describen a continuación.

2.1. Motores

El Observatorio Astronómico de la Universidad Tecnológica de Pereira cuenta con una cúpula o domo, el cual se desplaza en forma azimutal mediante el uso de un motor AC, además cuenta con 2 motores DC que permiten la apertura y cierre de la compuerta del domo, estos motores son accionados y alineados de forma manual.

2.1.1. Motor AC

Figura 1. Motor que genera el movimiento del domo

.

En la Figura 1 se presenta el motor AC que propicia el movimiento del domo, donde su eje central esta internamente unido a un piñón principal, el cual se conecta a una polea por medio de una cadeneta, y de esta manera al generar el movimiento, las demás poleas que sostienen el domo lo reflejan produciendo el desplazamiento del mismo.

Dicho Motor AC es manipulado por un Switch tipo codillo de tres estados ON/OFF/ON. La posición OFF permite el estado de reposo del motor y las posiciones ON generan el movimiento de su eje y a su vez el del domo hacia la derecha o izquierda según corresponda. Este motor posee las siguientes características:

Marca:	General Motors.
Modelo:	5KCP19MG429X
Potencia:	HP 1/20 (Caballos de fuerza)
Revoluciones:	RPM 1725/1450.

Voltaje: 115 Voltios.
Corriente: 0.95/1.15 Amperaje.
Frecuencia: 60/50 Hz.

2.1.2. Motores DC

Figura 2. Fuente de tensión para motores de apertura y cierre de compuertas de la cúpula.

La fuente de la Figura 2 se conecta a la toma eléctrica de 120V AC, permitiendo transformar dicha tensión de entrada a un valor aproximado de 5 Voltios, de esta manera se logra polarizar al valor nominal de voltaje de operación de los dos motores que permiten la apertura y cierre de compuertas de la cúpula, como se puede observar el la Figura 3, estos motores alcanzan niveles de corriente máxima de hasta 1.2A mientras se encuentran en operación.

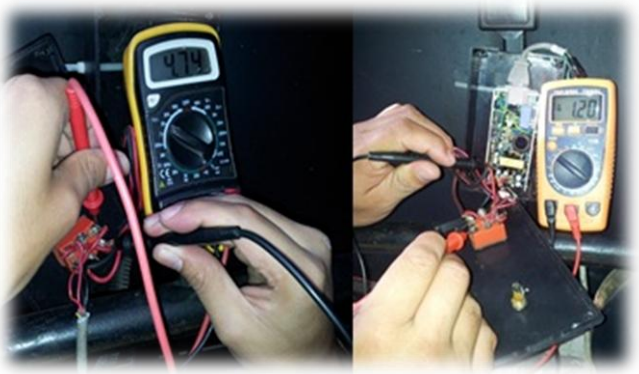


Figura 3. Tensión y Corriente máxima exigida por la carga (motores DC)

Dichos Motores DC son manipulados por un Switch tipo codillo de tres estados ON/OFF/ON. La posición OFF permite el estado de reposo de los motores y las posiciones ON generan la operación de los motores, quienes a su vez mueven las compuertas de derecha a izquierda o de izquierda a derecha según corresponda.

2.2. Telescopio MEADE16” LX200 – ACF

El Telescopio de la Figura 4 tiene un tamaño de 16” y su referencia es MEADE LX200 – ACF, permite realizar las observaciones y tomar las imágenes astronómicas en el observatorio. Actualmente es el principal instrumento de observación con el que cuenta el Observatorio Astronómico de la UTP. Cuenta con el software AutoStar II y su base de datos que contiene la referencia de más de 145000 objetos astronómicos.

Figura 4. Vista frontal del Telescopio MEADE 16” LX200 – ACF.

El telescopio MEADE 16” LX200 – ACF provee un amplio rango de resolución de imagen, el cuál libremente permite al observador tales comodidades como el estudio de los anillos del planeta Saturno que se encuentra a una distancia de 800 millones de kilómetros y el enfoque de distancias más lejanas del Sistema Solar como los antiguos cúmulos de estrellas, galaxias remotas y estrellas recientemente descubiertas que tienen planetas que orbitan alrededor de ellas.

Figura 5. Vista lateral inferior del Telescopio MEADE16” LX200 – ACF.

Sus principales características a nivel de hardware son: conector de voltaje DC para su alimentación, facilitando de esta manera el suministro de energía por medio de un adaptador en una salida de voltaje domiciliario de 115 a 120VAC, en la Figura 5 se puede observar el puerto de rotación para la base en la cual está apoyado el telescopio, conocido con el nombre de altazimuth, un puerto de salida de 12 VDC el cual alimenta los ventiladores ubicados en el tubo óptico, los cuales funcionan como filtro, evitando la entrada de partículas de polvo dentro del mismo, además funciona como un agente de enfriamiento en el interior del telescopio, posee puertos de comunicación RS232, Switch de encendido manual, puerto Handbox para comunicación vía telefónica para facilitar el control de movimiento en el equipo, salida de enfoque, la cual puede modificar la calidad en el acercamiento de la imagen tomada, puerto de rotación del equipo sobre el soporte o altazimuth.

En la Figura 6 se observa el sistema computarizado con el que cuenta el telescopio, que permite diferentes formas de control del telescopio, este se encuentra en la parte frontal inferior del mismo sobre la base que lo sostiene. Las formas de control utilizadas en el observatorio astronómico de la UTP son RS232 (Desde el PC local) y el control manual (HBX), dicha computadora se alimenta con 10 v DC.

Figura 6. Panel de control del telescopio MEADE 16” LX200 – ACF.

Desde su adquisición el telescopio MEADE 16” LX200 - ACF tiene una base de sostenimiento o trípode, pero debido a la altura donde se ubica el domo y sus compuertas de apertura y cierre, el telescopio MEADE 16” LX2000 - ACF se situó sobre un base metálica unida a una columna perteneciente al edificio.

2.3. Computador Local

Figura 7. Computador local que permite el control del telescopio MEADE 16” LX200 – ACF, mediante la aplicación AutoStar II

Este equipo de cómputo de la Figura 7 se encuentra conectado mediante conexión serial a la computadora del telescopio MEADE LX200, haciendo posible la interacción del usuario que administra el control de la visualización de los objetos celestes, mediante el software AutoStar II. Este equipo posee las siguientes características primarias:

Modelo:	HP COMPAQ dc5800 Small Form Factor
Sistema operativo:	Windows vista de 32 bits
Procesador:	Intel(R) Core(TM)2 Duo CPU E6550 Frecuencia de reloj 2.33GHz
Memoria RAM:	2.00 Gb
Tarjeta madre con referencia:	SPS-ASSY, TRAY, PCA, CATS-HOUNDS SFF
Fuente de poder de referencia:	SPS-BATTERY
Disco duro SATA de:	159 GB

El AutoStar II Database con más de 145000 objetos astronómicos referenciados está incluido sobre el Software de control del telescopio MEADE 16” LX200 – ACF, el cual se encuentra instalado en este computador local.

2.4. Cámara ST-2000 SBIG

En la Figura 8 se muestra una vista lateral de la cámara ST-2000 de la compañía SBIG con la que cuenta el Observatorio Astronómico de la Universidad Tecnológica de Pereira, esta cámara emplea el excelente CCD interlineal de Kodak, KODAK DIGITAL SCIENCETM KAI-2020M. Este CCD es un sensor de imagen

multimegapixel (1600 x 1200) de altas prestaciones diseñado para una amplia gama de aplicaciones en el campo científico. Sus píxeles de 7,4 micras con microlentes proporcionan una buena sensibilidad y capacidad de captación de electrones ofreciendo de esta manera un buen rango dinámico. Además, esta cámara incorpora para el seguimiento, un segundo CCD. Toda la electrónica de esta cámara viene integrada en un único cabezal. La conexión al ordenador se realiza a través de un puerto USB que ofrece una velocidad de transferencia de 400.000 píxeles por segundo.

Figura 8. Vista lateral de cámara ST-2000 SBIG.

2.5. Estación Meteorológica Weatherwise WS-1090

La estación meteorológica Weatherwise WS-1090 de la Figura 9 con la que cuenta el Observatorio de la Universidad Tecnológica de Pereira, es un instrumento que tiene una unidad base con pantalla táctil para usar en interiores y muestra importante información sobre el clima, de acuerdo a los datos proporcionados por los sensores que se encuentran al aire libre y envían dicha información de forma inalámbrica.

Figura 9. Estación Meteorológica.

Características básicas de la estación meteorológica Weatherwise WS-1090:

Frecuencia de transmisión:	915 MHz.
Distancia máxima de transmisión:	150 metros.
Almacenamiento de datos:	4080 Archivos en unidad base.
Visualización de datos:	Pantalla Touch –Screen y PC
Rango de temperatura:	- 40C a 65C (- 40F a 149F).
Rango de medición de humedad relativa:	10 % - 99 %.
Pronostico de tiempo:	Iconos del tiempo (soleado, nublado, lluvioso).

3

Observatorio en Línea

3.1. Esquema General

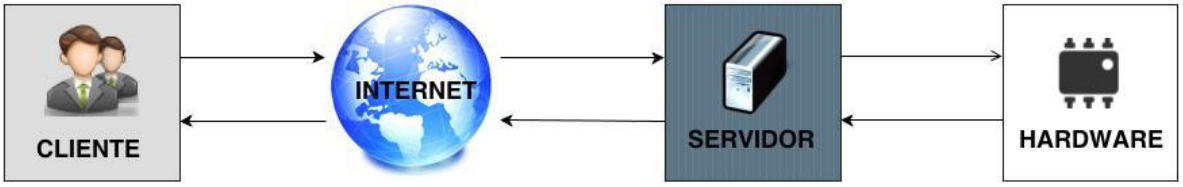


Figura 10. Descripción general del aplicativo de control vía web del observatorio astronómico de la Universidad Tecnológica de Pereira.

En el esquema de la Figura 10 se presenta el diseño general para permitir a cualquier usuario enfocado en el campo de la astronomía, con los permisos suficientes, controlar remotamente el observatorio astronómico ubicado en la Universidad Tecnológica de Pereira, el cual se divide en cuatro etapas principales: cliente, internet, servidor y hardware. Cada fase tiene una funcionalidad independiente que cumple un objetivo específico como es: controlar los instrumentos básicos para la observación astronómica por medio de dispositivos electrónicos como los micro controladores; administrar y procesar la información desde el cliente hasta al hardware y viceversa, y por ultimo proveer una aplicación web disponible para controlar el hardware.

3.1.1. Cliente

Como se observa en la Figura 11, el proceso de funcionalidad la provee el cliente o usuario final, que en este caso puede pertenecer a una institución académica, científica o bien ser un conocedor del tema de la astronomía.

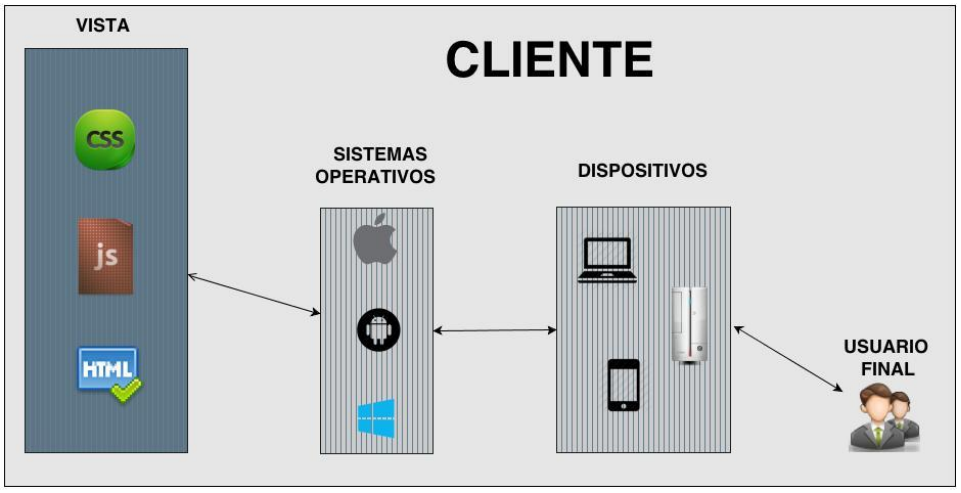


Figura 11. Descripción general de etapa cliente.

Capítulo 3. Observatorio en Línea

A nivel del usuario final el aplicativo le es solo visible lo que idealmente fue programado para su comodidad y fácil acceso, realmente las pantallas del aplicativo dependen de la arquitectura que compone lógicamente el procesamiento y tratamiento de los datos seleccionados por el usuario, por ello se refleja la integración de diversos servicios que conjuntamente satisfacen al usuario en el campo de interés. Las acciones del usuario con las pantallas del aplicativo se validan previamente para usarlo correctamente y dar a entender adecuadamente su funcionamiento usando la misma herramienta común conocida como navegador web.

El aplicativo provee su funcionalidad cuando el usuario por medio del navegador web hace la petición a una pantalla, la cual apunta a un servidor web quien devuelve a dicha petición la vista solicitada.

Las pantallas o vistas mostradas al usuario pertenecen a la parte del cliente, estas se diseñan a partir lenguajes que son conocidos como Javascript, tecnologías como CSS y la especificación de páginas web mundial conocida como HTML.

Dichas tecnologías de libre licenciamiento han permitido crear diferentes aplicaciones sobre la internet para cualquier tipo de usuario y tipo de necesidad, los usuarios usan navegadores web, los cuales interpretan la estructura de HTML y la traducen en forma de texto incluyendo objetos como imágenes, videos, sonidos para hacerla más entendible por el usuario final; no obstante, HTML es una información de muestra plana para el usuario por lo que los lenguajes de programación PHP y Javascript le proveen funcionalidad para permitir al usuario interactuar con la página de cierta manera catalogada como aplicación web, por lo tanto el uso de dichos lenguajes y tal estándar de la internet se han acomodado a la necesidad de diseñar el aplicativo orientado al mundo de la astronomía "*Control vía web del observatorio Astronómico de la Universidad Tecnológica de Pereira*".

Las páginas web conocidas también como "*vistas*" o "*pantallas*" del aplicativo en su diseño incrustan lenguaje Javascript para requerir funcionalidades dinámicas (Interacciones al usuario) y hojas de estilo para lograr una fachada elegante y agradable para el usuario, de esta manera, la interactividad con la página web es versátil y se ajusta a las necesidades, en este caso al campo de la astronomía. Dicho dinamismo se puede reflejar en imágenes animadas e información compartida.

3.1.2. Internet

La internet como red informática de comunicación internacional que permite el intercambio de información entre sus usuarios, es el medio utilizado para que los diferentes clientes interesados en utilizar el Observatorio Astronómico de la Universidad Tecnológica de Pereira puedan hacerlo desde cualquier parte del mundo donde se cuente con esta importante conjunto de redes descentralizadas (internet).

3.1.3. Servidor

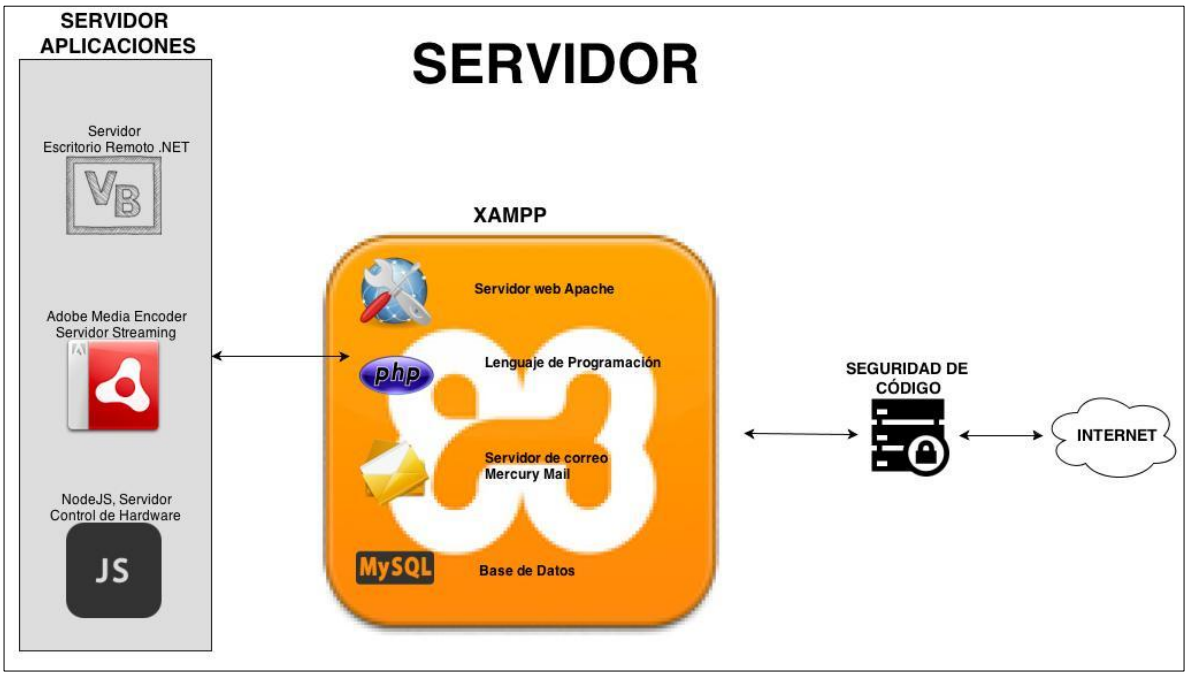


Figura 12 Descripción general de etapa servidor.

En la Figura 12 se describe la etapa de servidor, en esta etapa se administra y procesa la información ya sea del cliente o del hardware, para que sea visualizada por el usuario o para interpretarla de tal manera que pueda dar órdenes sobre el "*hardware"* y lograr proveer funcionalidades reales como abrir o cerrar compuertas, mover el domo, manipular software cerrados como el AutoStar o WeatherStation.

3.1.4. Hardware

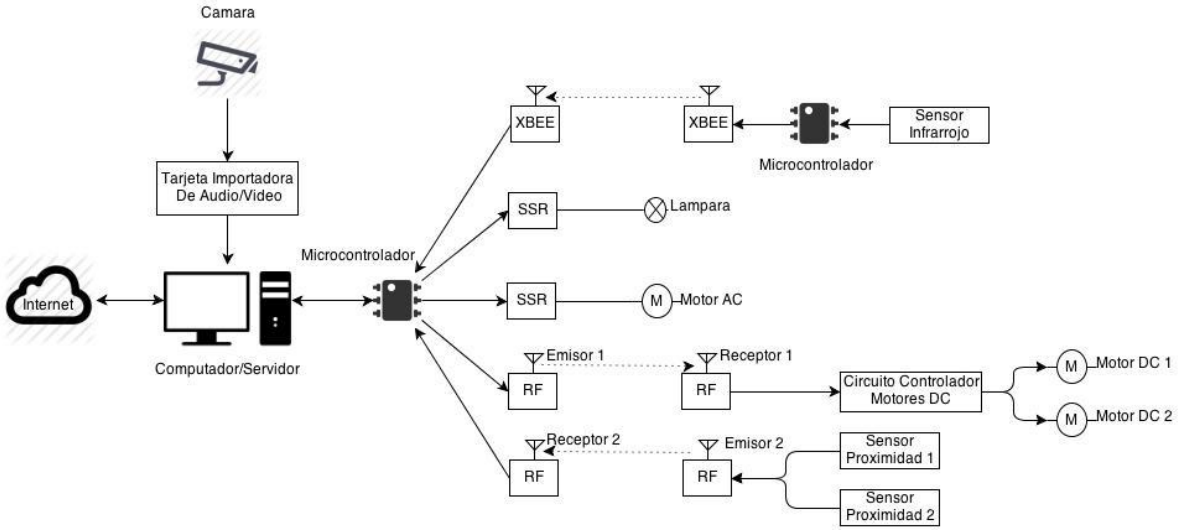


Figura 13. Diagrama general de hardware.

El control del hardware que se encuentra en el observatorio astronómico se logra mediante instrucciones dadas por el usuario desde el aplicativo web, que finalmente se traducen en la lógica necesaria para la ejecución de los servicios y en la acción determinada sobre las herramientas de observación o de hardware.

En la Figura 13 se describe la etapa del módulo hardware del aplicativo, esta etapa compete a la funcionalidad electrónica de diversos dispositivos enlazados entre sí para coordinar las acciones finales de apertura de las compuertas, rotación del domo, alineación de telescopio, control de encendido y apagado de una fuente de luz y adquisición de audio y video en streaming del interior del domo.

Este proceso se encuentra coordinado por un microcontrolador principal ATmega328 montado sobre una plataforma Arduino, el cual recibe y envía a un servidor las señales lógicas necesarias para control de los dispositivos electrónicos. Estos dispositivos igualmente se encuentran conectados a este microcontrolador enviando señales de estado mediante el uso de sensores y recibiendo señales de control para ejecutar las tareas que finalmente requiere el usuario. El funcionamiento de esta etapa se describe y amplia en este documento en el capítulo 4, Desarrollo de Hardware.

4

Desarrollo de Hardware

4.1. Control de Apertura y Cierre de Compuerta

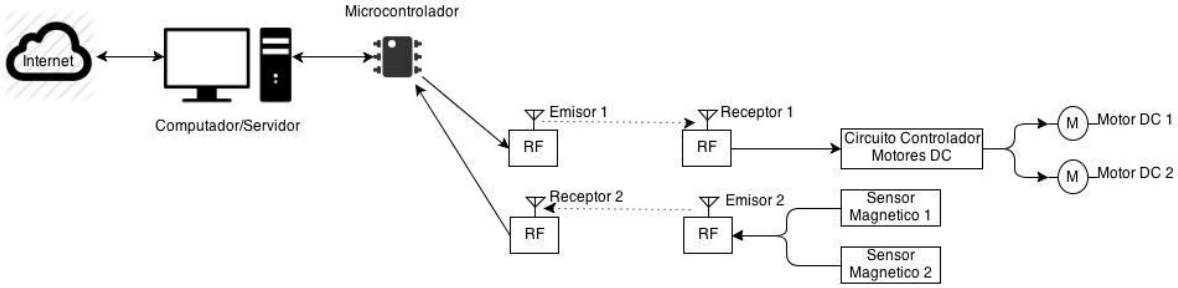


Figura 14. Diagrama general de control de compuerta.

En la Figura 14 se describe el diagrama general para la implementación del control de apertura y cierre de la compuerta del domo del Observatorio, donde fueron utilizados los siguientes componentes:

- **Módulo Arduino Uno:** Arduino es una plataforma de hardware libre, que está basada en un diseño que utiliza microcontroladores y una serie de hardware adicional el cual conecta el microcontrolador con las diferentes variables exteriores [7]. El microcontrolador utilizado es el ATMEL AVR el cual es uno de los más utilizados por su facilidad de uso y bajo costo en el mercado, permitiendo de esta manera el desarrollo de múltiples diseños en la industria actual [8], su entorno de desarrollo fue construido en base al lenguaje de programación Processing/wiring y el software de arranque conocido con el nombre de boot loader el cual está grabado en el microcontrolador. Este dispositivo permite ser programado mediante el uso de lenguaje C y admite el uso de instrucciones básicas.

Figura 15. Placa Arduino Uno

En la Figura 15 se observa como los microcontroladores utilizados se encuentran montados sobre una plataforma de Arduino, en la Figura 16 se detalla cómo esta plataforma se encuentra integrada a un shield Xbee de Arduino con además un Xbee Pro Serie 1 [9].

Figura 16. Módulo Arduino Uno con Shield Xbee y Xbee integrado.

- **Módulos de transmisión RF TLP434A:** En la Figura 18 se observa este módulo, que funciona como un transmisor inalámbrico de un tamaño pequeño, idealmente es utilizado como control remoto de la transferencia de datos a un objeto remoto [10]. Entre sus aplicaciones se encuentran los robots móviles, alarmas, control remoto y transferencia de datos inalámbricos. Esta unidad compacta opera desde 1.5V hasta 12V. Tiene un alcance de 150 metros al aire libre. La unidad se puede conectar directamente al codificador HT12E o similar. Con únicamente 4 pines, y su tamaño es de solo 13.3x10.3mm, la forma de transmitir es por Modulación por desplazamiento de amplitud, de esta forma los datos digitales se transmiten con modulación de amplitud, en la Figura 17 se describen sus principales características.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating supply voltage		2.0	-	12.0	V
Icc 1	Peak Current (2V)		-	-	1.64	mA
Icc 2	Peak Current (12V)		-	-	19.4	mA
Vh	Input High Voltage	Idata= 100uA (High)	Vcc-0.5	Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata= 0 uA (Low)	-	-	0.3	V
FO	Absolute Frequency	315Mhz module	314.8	315	315.2	MHz
PO	RF Output Power- 50ohm	Vcc = 9V-12V	-	16	-	dBm
		Vcc = 5V-6V	-	14	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

Notes : (Case Temperature = 25°C +- 2°C , Test Load Impedance = 50 ohm)

Figura 17. Características del módulo RF TLP434A

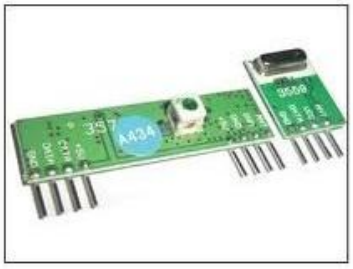


Figura 18. Módulos de transmisión RF TLP434A y RLP434A

- **Módulos de recepción RLP434A:** Este módulo se caracteriza por componerse de 8 pines y su tamaño es de 11.5x43.42mm y transmite de la misma forma como el emisor Amplitude Shift Keyink (ASK), en la Figura 19 se describen sus principales características.

Symbol	Parameter	Conditions	Min	Typ	Max	
Vcc	Operating supply voltage		3.3	5.0V	6.0	V
Itot	Operating Current		-	4.5		mA
Vdata	Data Out	Idata = +200 uA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics					
Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC	315, 418 and 433.92			MHz
Sensitivity	Pref		-110		dBm
Channel Width			+-500		Khz
Noise Equivalent BW			4		Khz
Receiver Turn On Time			5		ms
Operation Temperature	Top	-20	-	80	C
Baseboard Data Rate			4.8		KHz

Figura 19. Características del módulo RF RLP434A

- **Sensor o Interruptor Magnético:** Los interruptores magnéticos están conformados principalmente por láminas ferro-magnéticas construidas herméticamente en un encapsulado de cristal, dependiendo de su aplicación dicho encapsulado puede variar [11]. Las láminas están ubicadas a cierta distancia una de la otra de tal manera que en el momento que surge un campo magnético formado por un imán o por una bobina dicha fuerza mueva las láminas realizando el contacto en estas de tal manera que se realice la función de switch, por ende cuando la fuerza del campo magnético desaparece las láminas vuelven a su posición original, en la Figura 20 se visualiza un sensor o interruptor magnético de uso común.

Figura 20. Sensor o Interruptor Magnético.

- **Puente H:** Es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. Son ampliamente usados en robótica y como convertidores de potencia. Los puentes H están disponibles como circuitos integrados [12], pero también pueden construirse a partir de componentes discretos.

  Dentro del diseño de esta plataforma se utilizó el puente H L298, en la Figura 21 se describen los pines de este dispositivo.

  Características y Funciones

  - Amperaje máximo: 2A por canal
  - Voltaje de operación: de 4.5V a 46V
  - Señales de control: de 4.5 a 7V
  - Control: bidireccional del giro
  - Control de velocidad: Mediante técnica PWM
  - Motores simultáneos: 2

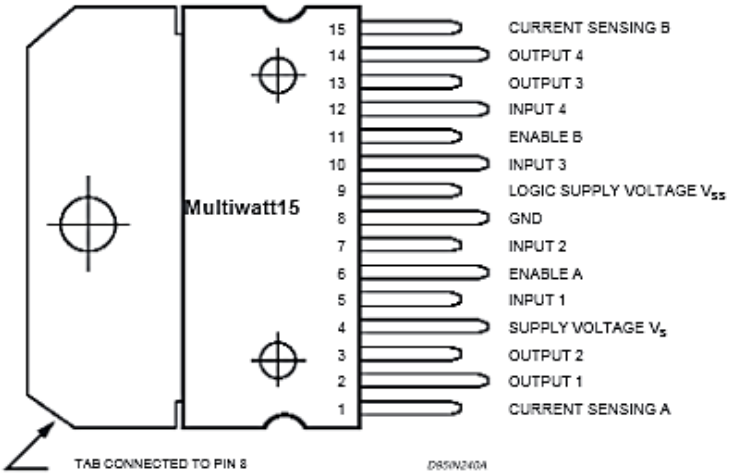


Figura 21. Diagrama de pines de puente H L298.

Para la apertura y cierre de la compuerta del domo del Observatorio Astronómico de la universidad Tecnológica de Pereira, se diseñó un sistema de control coordinado por un microcontrolador central Atmega montado sobre un módulo Arduino Uno, este microcontrolador realiza 2 tareas específicas relacionadas con la apertura y cierre de las compuertas del domo, la primera de ellas es recibir la información del estado lógico de los sensores magnéticos que puede variar como se describe en la Tabla 1 y enviarlos mediante el uso de los módulos RF hasta los pines 12 y 13 del microcontrolador, que se encuentran configurados como entradas digitales.

Estado Lógico Sensor Pin 12	Estado Lógico Sensor 2 Pin 13	Estado
0	0	Compuerta en proceso de apertura o cierre
0	1	Compuerta cerrada
1	0	Compuerta abierta
1	1	Estado lógico no posible

Tabla 1. Estados lógicos de sensores magnéticos de compuertas.

El segundo proceso consiste en enviar desde los pines 10 y 11 del microcontrolador configurados como salidas digitales, los datos lógicos necesarios para realizar las acciones enviadas por el usuario desde el aplicativo web. Estos datos lógicos son transferidos mediante el uso de los módulos RF hasta el circuito controlador de motores (Puente H), quien a su vez permite de acuerdo a la lógica que se muestra en la Tabla 2, el movimiento de los motores DC, lo que finalmente se traduce en la apertura y cierre de las compuertas del domo.

Pin 10	Pin 11	Motor 1	Motor 2	Acción
0	1	Gira a la derecha	Gira a la izquierda	Abre compuerta
1	0	Gira a la izquierda	Gira a la derecha	Cierra compuerta
0	0	Apagado	Apagado	Sin Acción

Tabla 2. Señales de estado de motores DC de compuertas.

El apagado de los motores y la posición final de la compuerta está controlado también por el estado de los sensores magnéticos instalados como se detalla en la Figura 22, de modo que estos se apagan de forma automática cuando se encuentran en la posición ideal que el usuario ha indicado desde el aplicativo web.

Figura 22. Sensor magnético instalado en compuerta del domo.

El control de las acciones realizadas por el usuario desde el aplicativo web, están determinadas por la operación del microcontrolador que contiene una lógica programada que se describe en el capítulo 4.5. Sistema Central (microcontrolador).

4.1.1. Circuitos RF emisor y receptor con señal de estado de compuertas

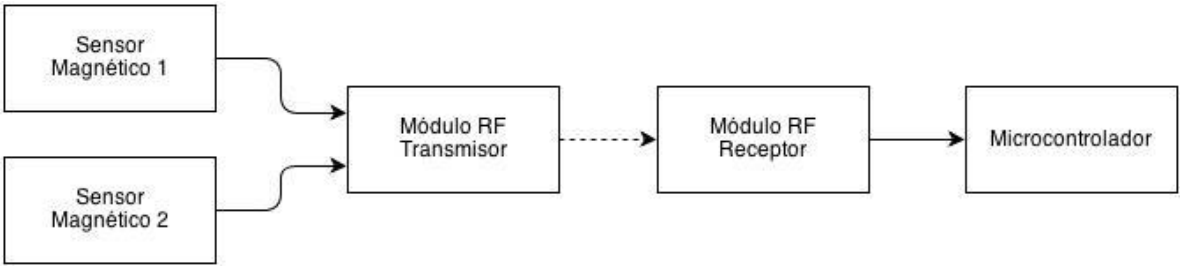


Figura 23. Diagrama de Conexión de circuito de estado de compuertas.

En la Figura 23 se describe el diagrama del circuito de control de apertura y cierre de las compuertas del domo del observatorio, que cuenta con 2 módulos de transmisión de señal RF, y 2 módulos de recepción. Estos módulos tienen una frecuencia de operación de 434MHz y fueron construidos en base a los integrados TLP434A y RLP434.

Un par de estos circuitos permiten enviar de forma inalámbrica las señales de estado lógico (Tabla 1) de las compuertas desde los sensores magnéticos hasta el microcontrolador y el otro par permiten la transmisión de la lógica necesaria (Tabla

2) para el encendido de los motores desde el microcontrolador hasta el puente H del circuito controlador de los motores DC.

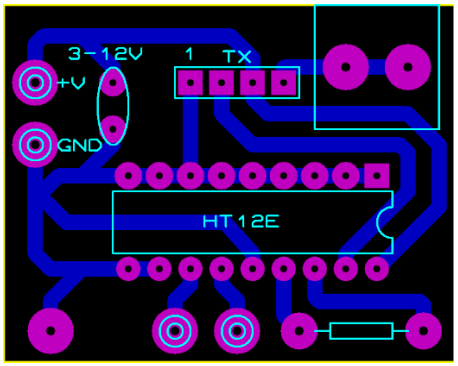


Figura 24. PCB del Circuito Emisor RF.

Se diseñaron los circuitos impresos para estos módulos como se muestran en las Figuras 24 y 25, mediante la utilización del software ISIS de Proteus y se obtuvo el diseño final como se aprecia en la Figura 26.

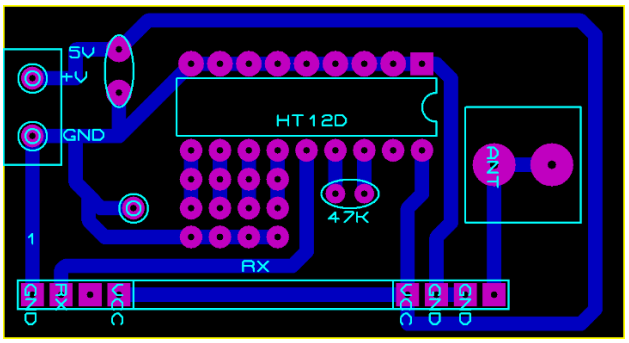


Figura 25. PCB del Circuito Receptor RF.

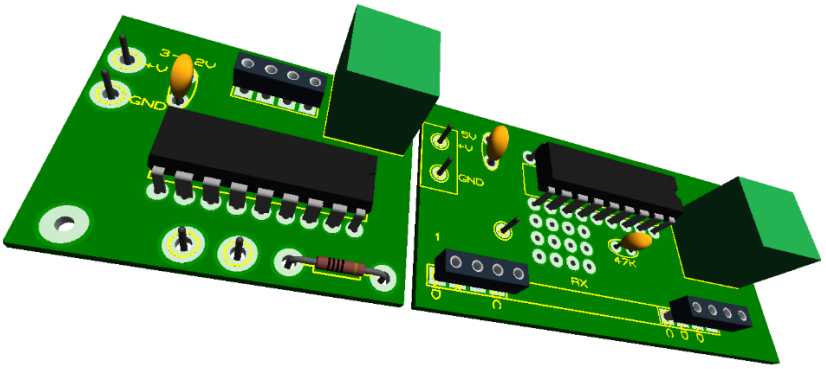


Figura 26. Diseño final de módulos RF.

En la Figura 27 se observa el estado final de la instalación de estos módulos, se sujetaron a cajas plásticas para proyectos electrónicos y los cables fueron cubiertos con coraza flexible de ¼.

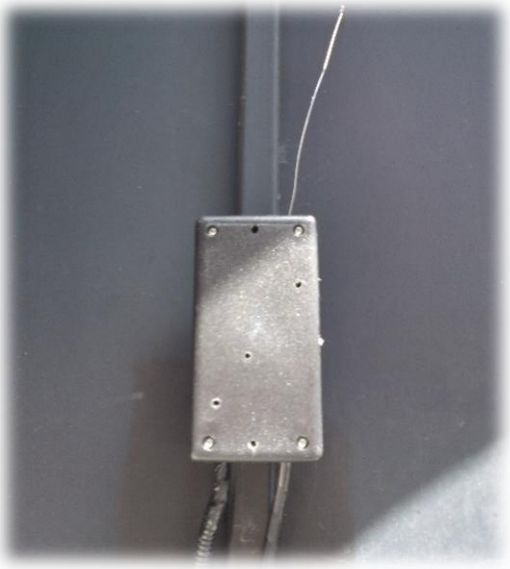


Figura 27. Módulo RF Instalado.

4.1.2. Circuito controlador de motores DC para cierre de compuertas

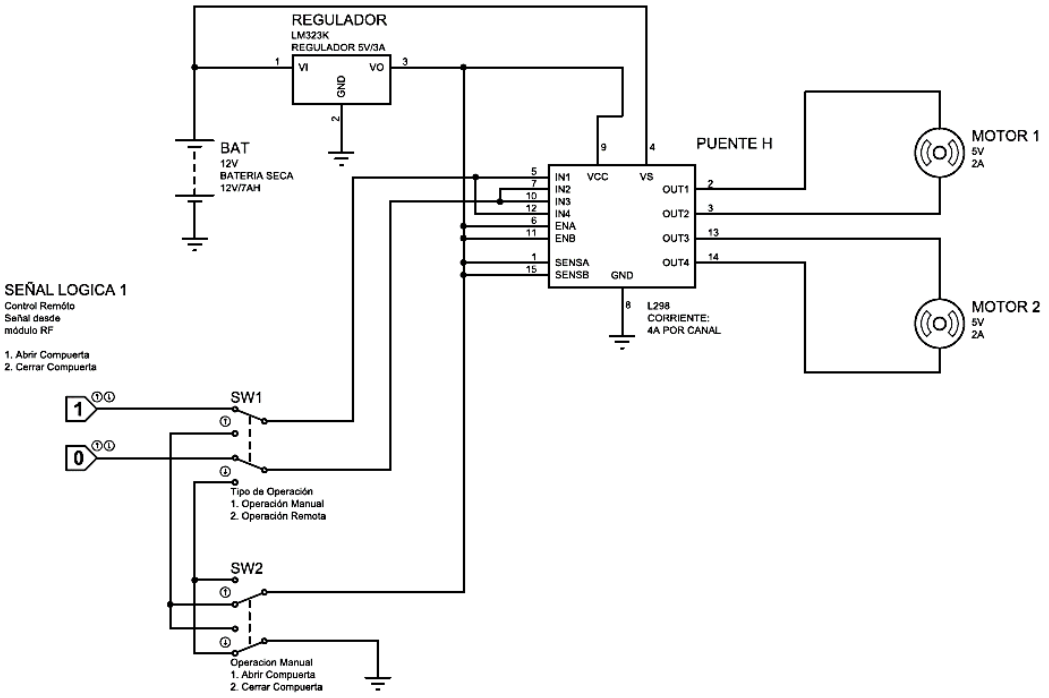


Figura 28. Circuito controlador de motores DC.

En la Figura 28 se describe el circuito de control de las compuertas del domo, que se diseñó teniendo en cuenta que sus circuitos no podrían estar conectados mediante cables o similares a ningún espacio diferente al domo, puesto que este gira de forma azimutal y debe ser electrónicamente independiente de los demás espacios del observatorio.

Considerando esta situación, se incluyó dentro del diseño del circuito una fuente de tensión de 12V, esta fuente es conocida comercialmente como batería seca 12V/7AH, además se agregó una fuente de carga para la batería, de modo que cuando esta se requiera puede ser conectada a una toma eléctrica de 120V AC, en la Figura 29 se observa la batería con su respectivo cargador, instalada dentro de una caja de proyectos.

Figura 29. Batería Seca 12V/7AH y su respectivo cargador en caja de proyecto.

Todo el circuito de control funciona con una tensión de 5V DC, esta tensión se logra mediante una fuente regulada de 12 a 5V construida en base al integrado LM 2596 de la Figura 30, el cual permite conectar una carga de hasta 3A en su terminal de salida.

Figura 30. Fuente regulada 12 a 5V DC.

El circuito de control de las compuertas del domo esta además constituido por un driver dual para motores DC o puente H de referencia L298, que se observa en la

Figura 31, este permite controlar de forma independiente los dos motores de las compuertas del domo, mediante las señales lógicas recibidas desde el microcontrolador de forma inalámbrica a través de los módulos de radiofrecuencia.

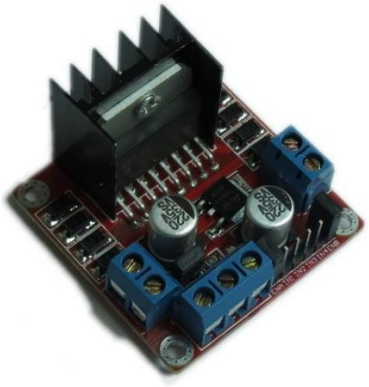


Figura 31. Puente H L298 para control de motores DC.

Como se visualiza en Figura 32, mediante el uso de interruptores tipo codillo, se incluyó en el circuito de control la posibilidad de elegir entre el control manual o el control web (SW1 de la Figura 28), y además se instalaron los conmutadores necesarios para el accionamiento y apagado de los motores en el caso de elección del control manual (SW2 de la Figura 28). De igual manera se incluyó un interruptor para permitir o no permitir el encendido de todos los componentes del circuito o colocar en modo carga la batería seca 12V/7AH.

Figura 32. Cajas contenedoras de circuitos con Switch de control.

4.2. Control de Rotación del Domo

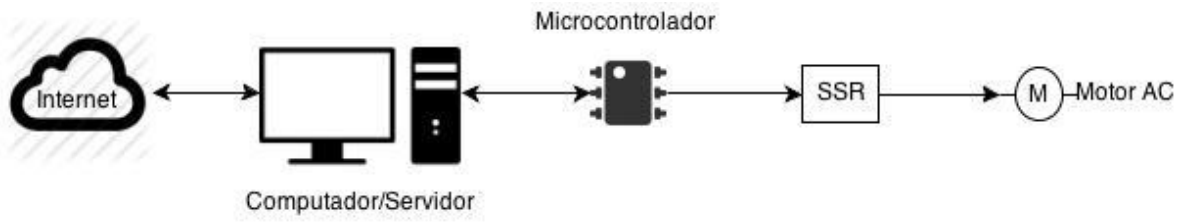


Figura 33. Diagrama de control de rotación del domo.

En la Figura 33 se describe el diagrama de control para la rotación del domo, donde se pueden visualizar los elementos utilizados en esta etapa, los dispositivos utilizados para este fin se describen a continuación.

Los Relés de Estado Sólido o SSR por sus siglas en inglés: SolidState Relays, son dispositivos ampliamente utilizados en la conmutación de cargas resistivas e inductivas con señales de control en DC o AC

Figura 34. Relé de Estado Solido SS440DA.

Como su nombre lo indica los Relés de Estado Utilizan componentes de estado sólido o semiconductores como los TRIACS y los SCRS, a diferencia de los relés electromagnéticos y contactores convencionales que en su construcción emplean bobinas y elementos mecánicos para realizar la conmutación.

型式 Type	SS-DA						SS-AA		
型號 Model	210DA 410DA	225DA 425DA	240DA 440DA	250DA 450DA	275DA 475DA	475DAR (SCR)	210AA 410AA	225AA 425AA	240AA 440AA
輸出額定電流(Arms) Rated load current	10A	25A	40A	50A	75A	75A	10A	25A	40A
工作電壓 Operating voltage	24~280Vac / 48~480Vac						24~280Vac / 48~480Vac		
峰值電壓 Non rep.peak voltage	600Vp / 800Vp						600Vp / 800Vp		
單週最大衝擊電流 1-cycle surge current	150A	250A	400A	500A	750A	750A	150A	250A	400A
OFF狀態漏電流(Max) Leakage current	3mA	8mA	8mA	8mA	8mA	8mA	3mA	8mA	8mA
ON狀態飽和電壓 On state voltage drop	1.6V/25°C						1.6V/25°C		
動作應答時間 Response time	ON<10ms , OFF<10ms						ON<20ms , OFF<20ms		

Figura 35. Especificaciones técnicas de relé de estado sólido SS440DA.

En la Figura 34 se aprecia una vista del SSR SS440DA, esta referencia es utilizada en el control de rotación del domo debido a que cuenta con las características eléctricas suficientes para realizar el control sobre el motor que permite la rotación de domo, en la Figura 35 se encuentran las especificaciones técnicas de este relé de estado sólido.

Este SSR recibe una señal de estado lógico desde el pin 9 del microcontrolador 1 para su operación. Cuando el usuario desde el aplicativo web da la instrucción de movimiento o giro de la cúpula, el microcontrolador pone en alto (Señal de 5V DC) el pin 9, el cual se encuentra conectado al INPUT de control del SSR, logrando así que este permita la circulación de corriente a través de la carga (motor AC). Para lograr este funcionamiento se realizaron las debidas instalaciones del SSR dentro de una caja plástica para proyectos, como se observa en la Figura 36.

El movimiento de la cúpula se da por terminado de forma automática, cuando se logre la alineación entre el telescopio y la compuerta de la cúpula, este proceso se describe en el capítulo 4.3 de este documento.

Figura 36. SSR instalado en caja de proyecto con conexiones.

El sistema de rotación de la cúpula también cuenta con la opción de control manual mediante 2 switch tipo codillo, uno de ellos permite seleccionar entre el control manual y el control web y el otro switch permite direccionar el giro hacia la izquierda o hacia la derecha del motor o el apagado del sistema como se describe en la Figura 37.

.

Figura 37. Caja de circuitos con Switch de control manual.

4.3. Alineación de Telescopio y Compuerta del Domo

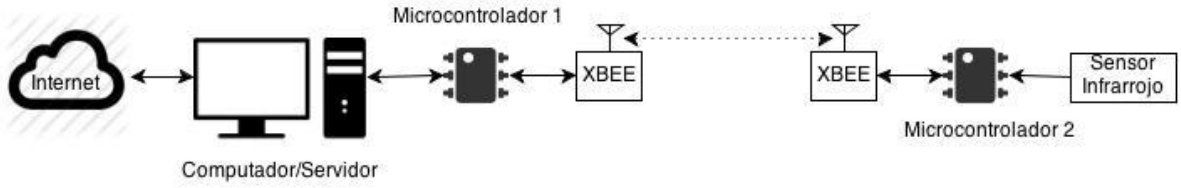


Figura 38. Diagrama del sistema de alineación entre telescopio y domo.

En la Figura 38 se describe mediante un diagrama el sistema de alineación entre el telescopio y el domo, para la implementación de esta etapa se utilizaron 2 módulos de comunicación XBEE, un segundo módulo Arduino con microcontrolador Atmega y un sensor infrarrojo, estos componentes se describen a continuación:

- **XBee:** Los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 [13] para crear redes FAST POINT-TO-MULTIPOINT (punto a multipunto); o para redes PEER-TO-PEER (punto a punto). Fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible. Por lo que básicamente XBee es propiedad de Digi basado en el protocolo Zigbee, En la Figura 39 se puede apreciar una fotografía de un módulo XBee Pro de la serie 1 utilizado en esta etapa.

Figura 39. XBee Pro S1.

- **Sensor Infrarrojo:** Este tipo de sensor está compuesto por un sensor receptor de rayos infrarrojos, el cual está compuesto por un fototransistor o fotodiodo. El circuito ubicado a la salida del circuito usa la señal recibida del receptor para realizar una amplificación en la salida y adaptarla a la aplicación, dicha señal transmitida puede ser codificada para facilitar la identificación de la señal enviada de la transmitida, además de identificar los diferentes dispositivos que se encuentren en un rango cercano.

  Uno de las categorías de este tipo de sensores son los sensor réflex, este tipo de sensor está compuesto por un sensor emisor y receptor en un solo componente, en el cual el rayo infrarrojo emitido es el encargado de realizar la detección de los objetos por una interrupción de dicho rayo emitido en el sensor de recepción, debido a esto la detección del objeto no se ve afectada por la coloración del mismo. La ventaja de las barreras réflex es que el cableado es en un solo lado, a diferencia de las barreras emisor-receptor que es en ambos lados.

  Un ejemplo de este tipo de sensor es el Sharp GP2Y0A02YK0F que se visualiza en la Figura 40 y es el utilizado en esta etapa de control.

Figura 40. Sensor Infrarrojo GP2Y0A02YK0F.

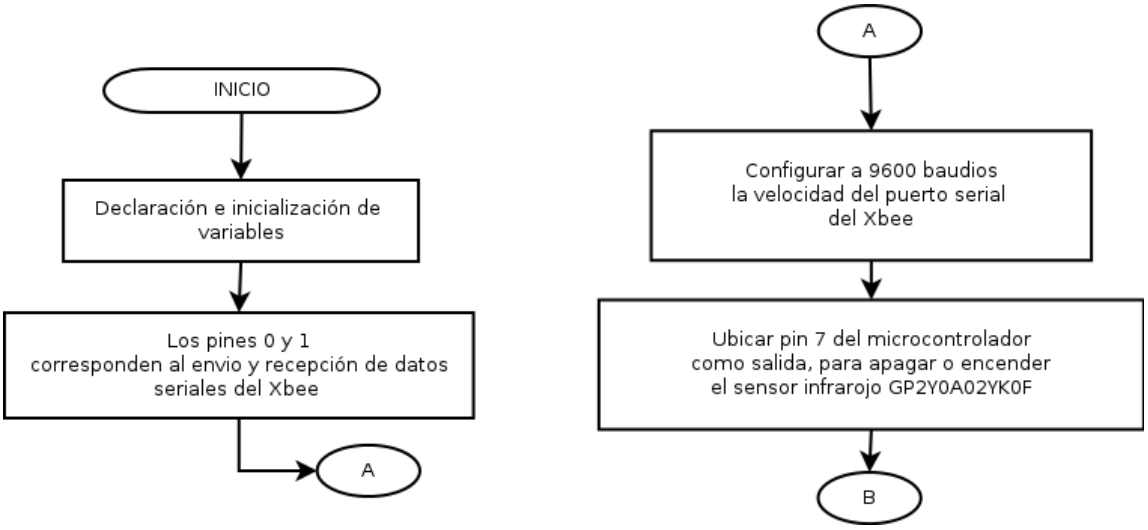
Este dispositivo es una unidad de sensor de medición de distancia, compuesto de una combinación integrada de PSD (detector sensible a la posición), emisión IRED (infrarrojos diodo) y el circuito de procesamiento de señales [14]. La variedad de la reflectividad del objeto, la temperatura

ambiental y la duración de funcionamiento no son influenciados fácilmente a la detección de distancia debido a la adopción del método de triangulación. Este dispositivo da salida a la tensión correspondiente a la distancia de detección. Así que este sensor también se puede utilizar como un sensor de proximidad.

Características:

Distancia de rango de medición:	20 a 150 cm.
Tipo de salida:	analógica.
Tamaño del paquete:	29.5 × 13 × 21,6 mm.
Corriente de consumo:	Typ. 33 mA.
Tensión de alimentación:	4,5 a 5,5 V

La alineación del telescopio se realiza mediante una instrucción dada por el usuario desde el aplicativo web, esta instrucción al llegar al microcontrolador 1 (Atmega328P) es enviada al SSR de control del motor AC para la rotación del domo, como se explicó en el capítulo 4.3 de este documento. De forma paralela, el sensor infrarrojo GP2Y0A02YK0F fijado sobre el centro del telescopio, envía valores comprendidos entre 0 y 1024 que indican en nivel de reflexión de la luz infrarroja sobre la superficie a la que este apuntando el led emisor. Esta información es procesada por el microcontrolador 2 (Atmega328P), donde mediante la lógica que se expresa en el diagrama de flujo que se describe en la Figura 41, se determina si el telescopio se encuentra alineado con la compuerta de la cúpula o aun no lo está. El código completo puede ser visualizado en el Anexo 1.



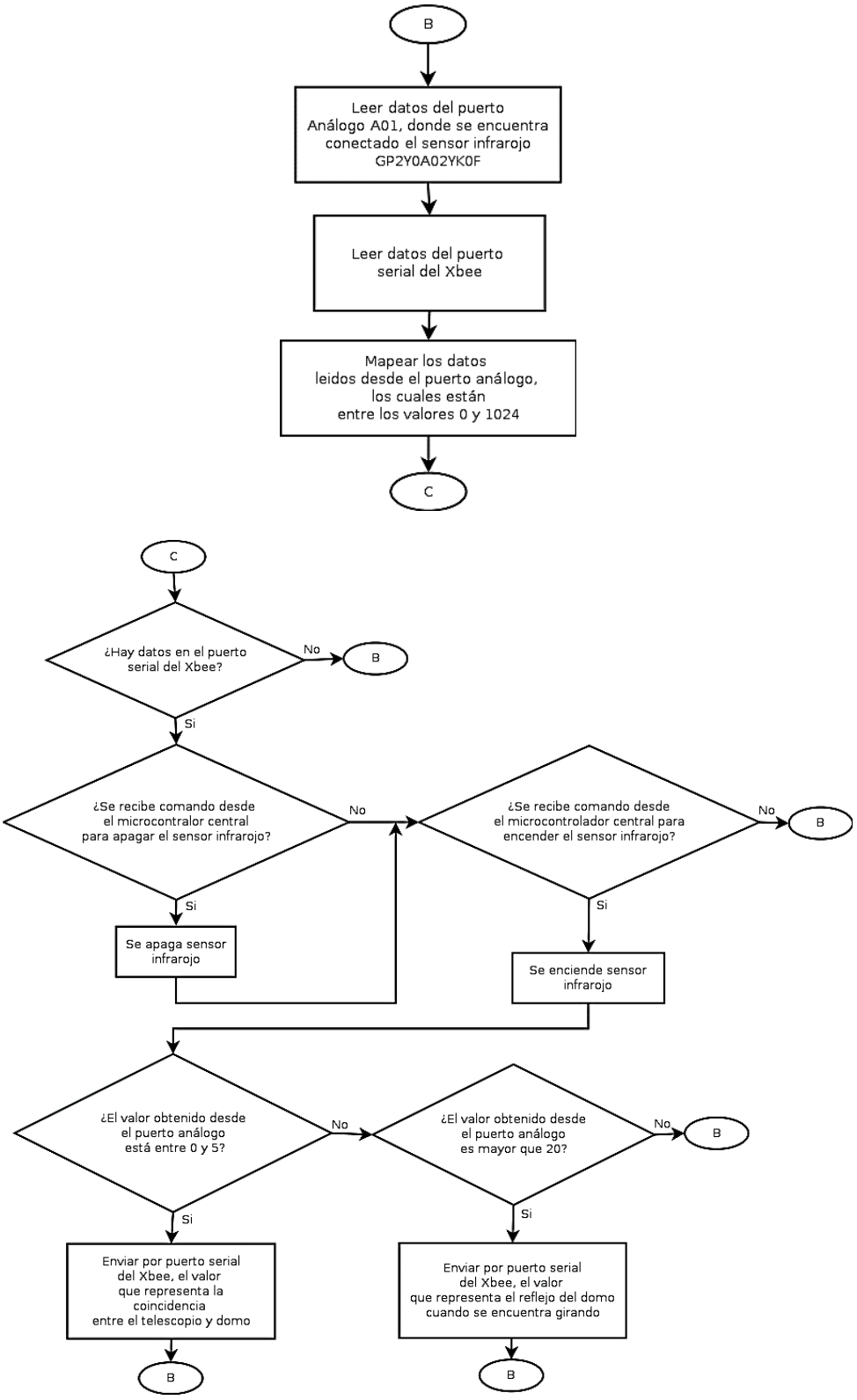


Figura 41. Diagrama de flujo de Microcontrolador 2.

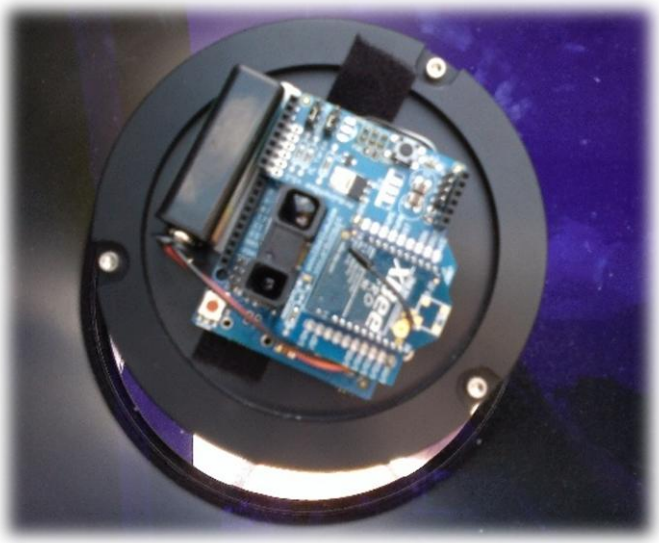


Figura 42. Módulo Arduino Uno con microcontrolador 2, sensor infrarrojo y XBEE fijado en centro del telescopio.

En la Figura 42 se aprecia como el módulo Arduino Uno con el respectivo shield, el dispositivo XBee y el sensor infrarrojo son montados sobre el centro del telescopio, de modo que el sensor siempre este apuntando en la dirección hacia la que apunta el telescopio y permita reconocer cuando se encuentra alineado con las compuertas del domo.

Los datos recibidos desde el sensor por el microcontrolador son mapeados o procesados en un rango menor para facilitar la lógica de funcionamiento y son guardados en una variable denominada sensorValue, si esta variable se encuentra en un rango entre 0 a 5 el microcontrolador 2 imprime por su puerto serial el valor “1”, si se encuentra en un valor mayor o igual a 20 el microcontrolador 2 imprime por el puerto serial el valor “0”, es decir, si la variable sensorValue se encuentra en un valor menor a 5, esto indica que el sensor y a su vez el telescopio se encuentra alineado con la compuerta, ya que la reflexión de la luz infrarroja emitida sobre el domo es mínima para este caso, pero si por el contrario la variable se encuentra en un valor mayor o igual a 20, se indica que el telescopio aún no se encuentra alineado con la compuerta.

Los valores “0” y “1” que establecen el estado de alineación entre la compuerta y el telescopio son enviados de forma inalámbrica mediante el uso de un módulo de comunicación XBEE al microcontrolador 1. Una vez llega esta información a este microcontrolador, la información es procesada y se da la instrucción al SSR del motor AC para que se detenga y envía el estado de detención al microcontrolador 2 para que apague el sensor infrarrojo y evite el consumo innecesario de energía de la batería, quedando además a la espera de una nueva posible alineación por parte del usuario conectado desde la plataforma web.

Para lograr la comunicación de los módulos XBEE, se configuraron previamente mediante el software X-CTU como se visualiza en la Figura 43, teniendo en cuenta los valores o parámetros de configuración descritos en la Tabla 3.

XBEE 1	XBEE 2
DH 13A200	DH 13A200
DL 4076E267	DL 4076E26E
MY AAAA	MY AAAA
SH 13A200 (por defecto)	SH 13A200 (por defecto)
SL 4076E26E (por defecto)	SL 4076E267 (por defecto)
CE 1-COORDINADOR	CE 0-End Device

Tabla 3. Valores de configuración de módulos XBEE.

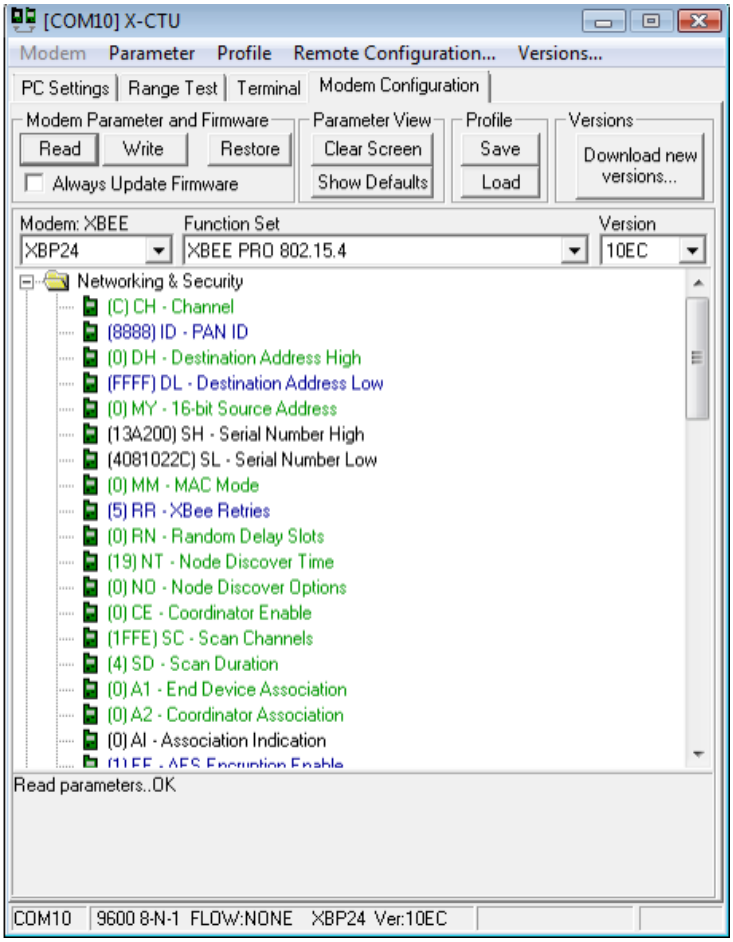


Figura 43. Software X-CTU en terminal de configuración de módulos XBEE.

4.4. Control de Encendido y Apagado de Fuente de Luz

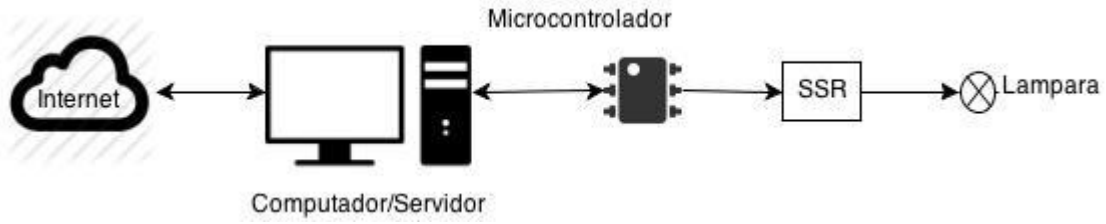


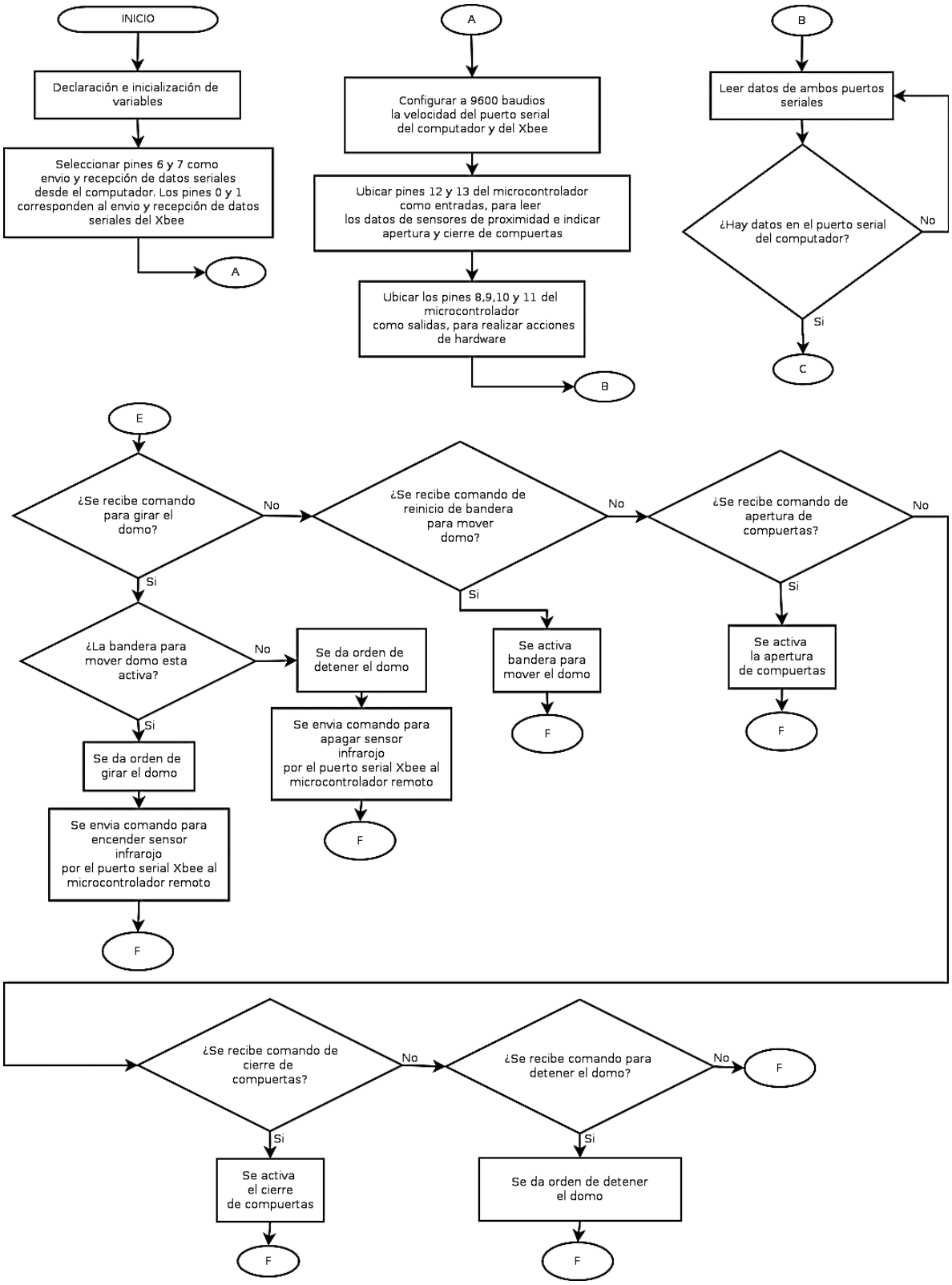
Figura 44. Diagrama de control de fuente de luz.

En la Figura 44 se describe el diagrama de control de encendido y apagado de una fuente de luz o luminaria en el observatorio, para esta etapa se utilizó un rele de estado solido SS440DA [15], el cual recibe las señales de control desde el microcontrolador 1 (Atmega328P), estas señales de control son inicialmente enviadas por el usuario, haciendo las peticiones desde la página web, En la Figura 45 se observan los dispositivos instalados al interior de la cúpula del observatorio.

Figura 45. Dispositivos instalados para control de luminaria.

4.5. Sistema Central

El procesamiento de estados lógicos de la plataforma de control vía web del Observatorio Astronómico, está centralizado en el microcontrolador 1, del cual se ha hecho mención a lo largo de este capítulo, este se encarga, mediante una conexión Serial RS232, de recibir la petición del usuario desde el servidor, de interactuar con los dispositivos de control y los sensores anteriormente señalados en este capítulo y de enviar el estado de los dispositivos al aplicativo web. La lógica de funcionamiento se resume en el diagrama de flujo de la Figura 46.


INICIO
Declaración e inicialización de variables
Seleccionar pines 6 y 7 como envio y recepción de datos seriales desde el computador. Los pines 0 y 1 corresponden al envio y recepción de datos seriales del Xbee
A
A
Configurar a 9600 baudios la velocidad del puerto serial del computador y del Xbee
Ubicar pines 12 y 13 del microcontrolador como entradas, para leer los datos de sensores de proximidad e indicar apertura y cierre de compuertas
Ubicar los pines 8,9,10 y 11 del microcontrolador como salidas, para realizar acciones de hardware
B
B
Leer datos de ambos puertos seriales
¿Hay datos en el puerto serial del computador?
No
Si
C
E
¿Se recibe comando para girar el domo?
No
Si
¿La bandera para mover domo esta activa?
No
Si
Se da orden de detener el domo
Se envia comando para apagar sensor infrarojo por el puerto serial Xbee al microcontrolador remoto
F
Se da orden de girar el domo
Se envia comando para encender sensor infrarojo por el puerto serial Xbee al microcontrolador remoto
F
¿Se recibe comando de reinicio de bandera para mover domo?
No
Si
Se activa bandera para mover el domo
F
¿Se recibe comando de apertura de compuertas?
No
Si
Se activa la apertura de compuertas
F
¿Se recibe comando de cierre de compuertas?
No
Si
Se activa el cierre de compuertas
F
¿Se recibe comando para detener el domo?
No
F
Si
Se da orden de detener el domo
F

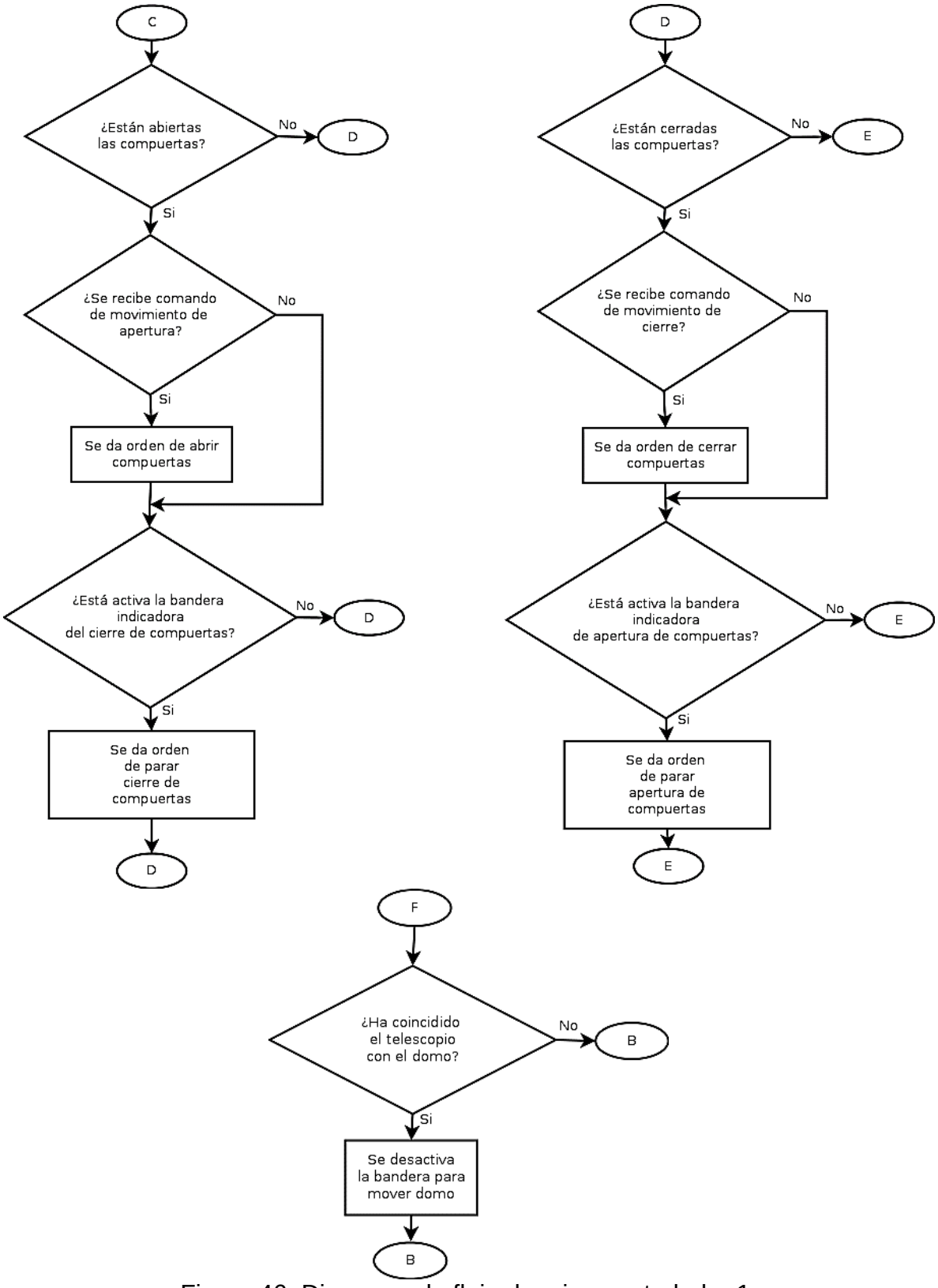


Figura 46. Diagrama de flujo de microcontrolador 1.

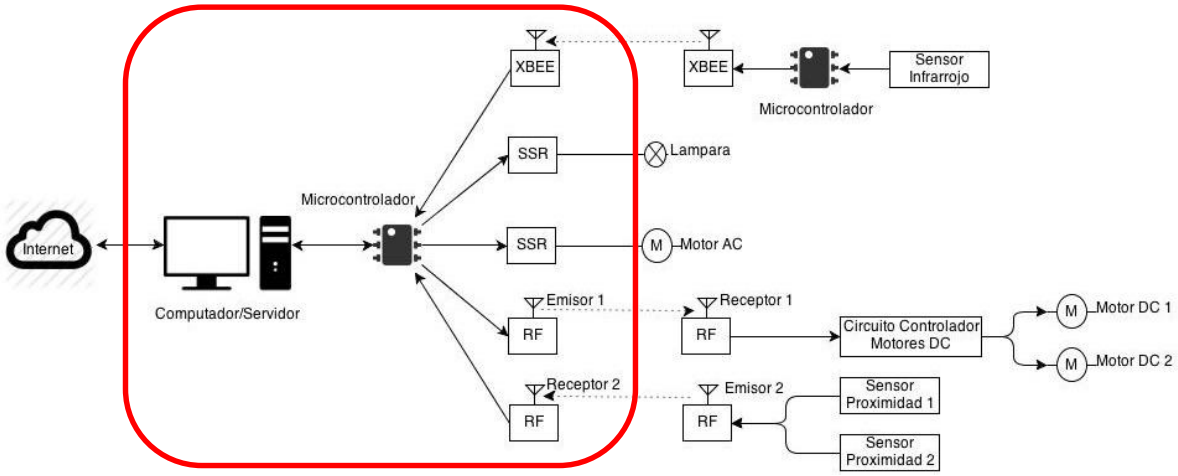


Figura 47. Sistema Central dentro del diagrama general.

En la Figura 47 dentro del cuadro, se muestran los dispositivos que se encuentran vinculados directamente con el microcontrolador, estos dispositivos están conectado al microcontrolador 1 del sistema, como se resume en la Tabla 4.

PIN	Configuración	Dispositivo	Medio de comunicación	Uso
13	Entrada digital	Sensor de proximidad 1	Modulo RF 2	Identifica si la compuerta está abierta.
12	Entrada digital	Sensor de proximidad 2	Modulo RF 2	Identifica si la compuerta está cerrada.
11	Salida digital	Puente H L298	Modulo RF 1	Señal de control de apertura de compuerta.
10	Salida digital	Puente H L298	Modulo RF 1	Señal de control de cierre de compuerta.
9	Salida digital	SSR motor AC	Cableada	Señal de control de encendido y apagado de motor AC.
8	Salida digital	SSR luminaria	Cableada	Señal de control de encendido y apagado de luminaria.
1	Serial RS232 - TX	Microcontrolador 2 con sensor infrarrojo	XBEE	Señal de control para el encendido del sensor y transmisión de datos desde microcontrolador 2.

0	Serial RS232 - RX	Microcontrolador 2 con sensor infrarrojo	XBEE	Señal de posición de alineamiento entre telescopio y compuerta de la cúpula.
6	Salida digita - TX	Computador - Servidor	Cableada	Transmite información desde el microcontrolador hacia el servidor.
7	Entrada digital -RX	Computador - Servidor	Cableada	Transmite información desde el servidor hacia el Microcontrolador.

Tabla 4. Distribución de pines del microcontrolador 1.

El módulo Arduino en el cual se encuentra instalado el microcontrolador 1, cuenta con un solo puerto serial. Es en este puerto donde se recibe la información proveniente del microcontrolador 2, al intentar transmitir la información del servidor o computador hacia el microcontrolador 1, utilizando el mismo puerto serial, se presentaron bloqueos, por este motivo se incluyó un nuevo puerto serial utilizando dos pines digitales del microcontrolador (6 y 7) mediante una librería desarrollada por Atmel, esta librería es conocida como SoftwareSerial [16].

Para lograr una adecuada implementación de este puerto serial se utilizó un cable USB-Serial RS-232, el cual es conectado a uno de los puertos USB del computador y mediante el uso del integrado Max-232 [17] se logró hacer el acople de señales entre el cable serial RS-232 y las señales TTL del microcontrolador, mediante la construcción del circuito que se describe en la Figura 48.

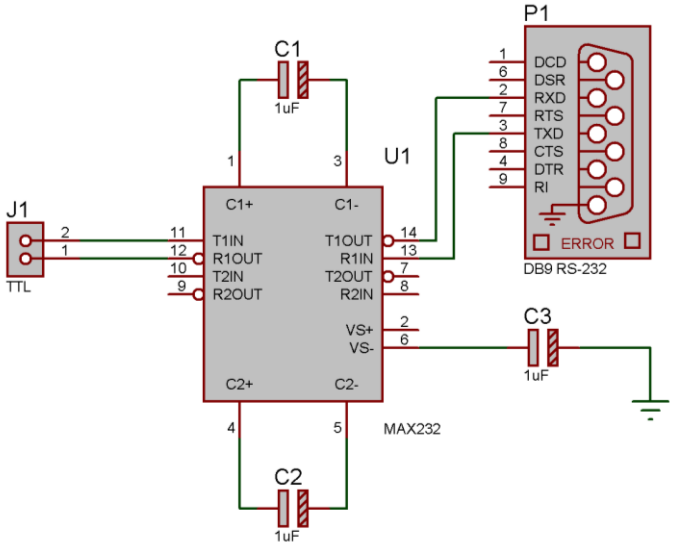


Figura 48. Esquema de circuito electrónico RS-232/TTL

El código correspondiente al sistema central (microcontrolador 1) se encuentra en el anexo 2.

4.6. Adquisición de Audio y Video en Streaming

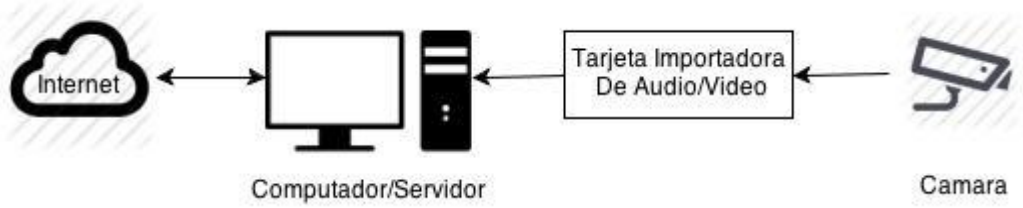


Figura 49. Diagrama de adquisición de audio y video para trasmisión en streaming.

En la Figura 49 se describe el diagrama de conexiones y dispositivos utilizados para la adquisición de video y transmisión del mismo mediante tecnología streaming.

La tecnología de streaming se utiliza para optimizar la descarga y reproducción de archivos de audio y video que se alojan o transmiten en vivo desde un servidor.

El streaming funciona de acuerdo los siguientes pasos:

- **Conexión con el servidor**. El reproductor cliente conecta con el servidor remoto y éste comienza a enviarle el archivo.
- **Buffer**. El cliente comienza a recibir el fichero y construye un buffer o almacén donde empieza a guardarlo.
- **Inicio de la reproducción**. Cuando el buffer se ha llenado con una pequeña fracción inicial del archivo original, el reproductor cliente comienza a mostrarlo mientras continúa en segundo plano con el resto de la descarga.
- **Caídas de la velocidad de conexión**. Si la conexión experimenta ligeros descensos de velocidad durante la reproducción, el cliente podría seguir mostrando el contenido consumiendo la información almacenada en el buffer. Si llega a consumir todo el buffer se detendría hasta que se volviera a llenar.

El streaming puede ser de dos tipos dependiendo de la tecnología instalada en el servidor:

- **Descarga progresiva**. Se produce en servidores web que disponen de Internet Information Server (IIS), Apache, Tomcat, etc. El archivo de vídeo o audio solicitado por el cliente es liberado por el servidor como cualquier otro archivo utilizando el protocolo HTTP. Sin embargo, si el archivo ha sido especialmente empaquetado para streaming, al ser leído por el reproductor cliente, se iniciará en streaming en cuanto se llene el buffer.
- **Transmisión por secuencias**. Se produce en servidores multimedia que disponen de un software especial para gestionar más óptimamente el

streaming de audio y vídeo: Windows Media Server, Flash Communication Server, etc. Para lograr la publicación en streaming del video interno de la cúpula del observatorio, se utilizó una transmisión por secuencias mediante aplicaciones de la compañía Adobe, que se tratan en el capítulo 5.1.6.

Para la adquisición del video se utilizó una cámara de seguridad genérica con conexión RCA y con leds infrarrojos para mejorar la visión en situaciones de poca luz al interior de la cúpula del observatorio, este dispositivo se visualiza en la Figura 50.

Figura 50. Cámara de seguridad y tarjeta importadora con conexión RCA.

Esta cámara se conecta a un dispositivo de adquisición de video o tarjeta importadora de video, la cual se encarga de procesar y enviar los datos de la cámara, hacia el equipo de cómputo utilizado como servidor, mediante una conexión USB.

La tarjeta capturadora utilizada es de marca Encore y su referencia es ENUTV-4, sus principales características se pueden observar en la Tabla 5.

TV format	Worldwide (except SECAM LL)
Resolution	Up to 720*480 (NTSC) or 720*576 (PAL)
Video Compression	MPEG-1, MPEG-2 and MPEG4(XVID)
RF Input	(75 ohm impedance, F or PH type) (By Mode)
TV input	(75 ohm impedance, F or PH type)
Video input	RCA video input / S-Video input
USB	USB connector Type B
Audio input	Phone jack
Dimension	107 x 77 x 25 mm
Certification	FCC, CE, RoHS

Tabla 5. Características de cámara utilizada para el streaming.

5

Desarrollo de Software

5.1. Configuración de Programas y Servidores

Para el adecuado funcionamiento de las diferentes aplicaciones y tecnologías usadas en la plataforma de control vía Web del Observatorio Astronómico, se hizo necesaria la instalación y configuración de varios programas y componentes que se describen a continuación:

5.1.1. Instalación de Xampp

Xampp es conocido como un distribuidor de instalación de un conjunto de aplicaciones como sistema de gestión de base de datos mySql [18], intérprete de lenguajes script como PHP y PERL, el servidor web Apache y el servidor Mail Mercury; conjuntamente permite integrar estas aplicaciones para conformar una funcionalidad común consistente en la publicación de páginas web dinámicas, brindando acciones sobre conexión al motor de base de datos y permitiendo procesar la información enviada desde el cliente (Navegador web).

Con el fin de tener integrado en el equipo de cómputo utilizado como servidor web, una herramienta o conjunto de aplicaciones mínimas para el funcionamiento de este, se realiza la respectiva instalación del paquete Xampp en su versión 3.1.0.3.1.0 como se muestra en la Figura 51.

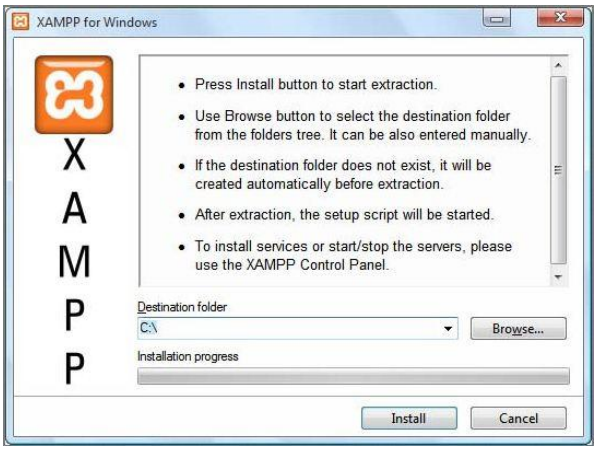


Figura 51. Inicio de instalación de Xampp en la unidad principal del sistema.

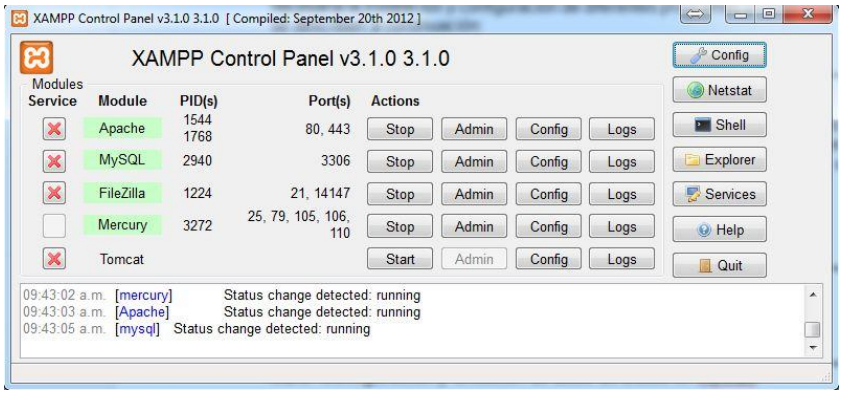


Figura 52. Panel de control de Xampp.

Al finalizar la instalación de acuerdo con la Figura 52, se verifica el correcto funcionamiento de cada una de las aplicaciones o servicios, identificando además el apropiado uso de puertos para cada uno de ellos, en especial para los módulos Apache, MySQL y Mercury.

5.1.2. Configuración de Mercury

Luego de la instalación del Xampp se procede con la configuración del servidor de correo Mercury, en la Figura 53 se muestra como ingresar al panel de configuración de este módulo en el que se agregan o modifican cada uno de los campos de acuerdo a las necesidades.

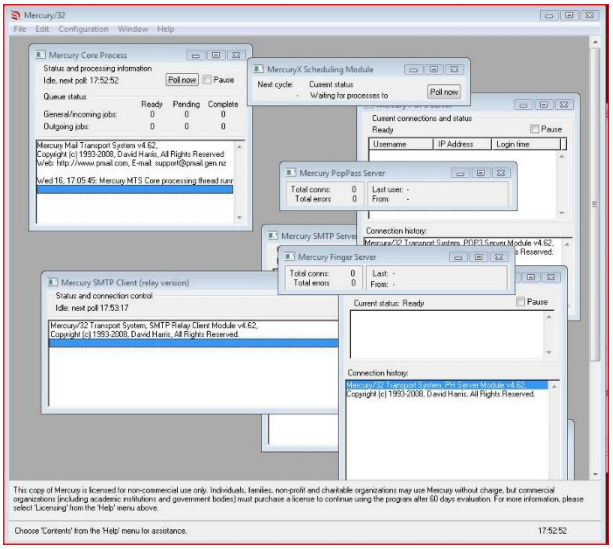


Figura 53. Panel de configuración del Módulo Mercury.

A continuación se describe cada uno de los pasos realizados para lograr una configuración adecuada en el módulo Mercury:

- En la ruta Configuration/Protocol Modules se desactiva “MercuryB HTTP web server” y “Mercury IMAP4rev1 server”. Para enviar emails a correos externos se desactiva “MercuryE SMTP end-to-end delivery client” y en cambio se activa “MercuryC SMTP relaying client”.

- En la ruta Configuration/Mercury core module en la pestaña General. En “internet name for this system” se agrega el dominio al cual apunta el servidor, que para este caso es 127.0.0.1.

- La Figura 54 describe la configuración del protocolo SMTP para los emails salientes en la ruta Configuration/MercuryS SMTP Server. Sobre la pestaña General, en el campo “Announce myself as” se inserta el nombre “Observatorio Astronómico UTP”. Se adiciona el TCP/IP port como 25, que es el del SMTP. En “IP interface to use” se agrega la dirección 127.0.0.1.

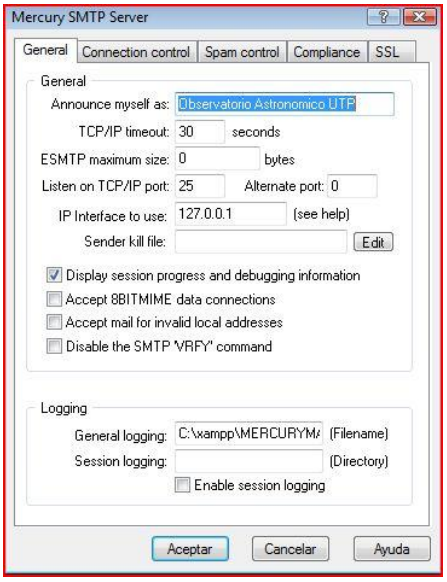


Figura 54. Configuración de SMTP en módulo Mercury.

- En la Figura 55 se muestra la configuración del protocolo POP3 del Mercury en la ruta Configuration/MercuryP POP3 Server. En la pestaña General se agrega como TCP port el 110 y la “IP interface to use” en 127.0.0.1.

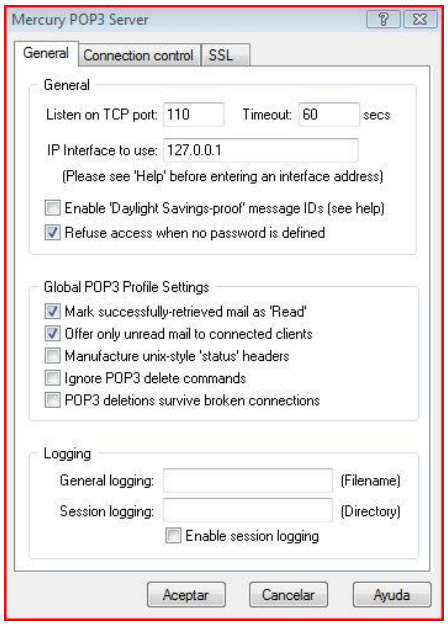


Figura 55. Configuración de POP3 en módulo Mercury.

- En la Figura 56 se describe la configuración del cliente para el protocolo SMTP del Mercury en la ruta Configuration/MercuryC SMTP Client. Para enviar emails al exterior se adicionan los datos de un correo exterior de Gmail. En “Smart host name” se agrega smtp.gmail.com. El puerto se configura como el 587 y se elige la opción STARTTLS que es el soportado por Gmail.

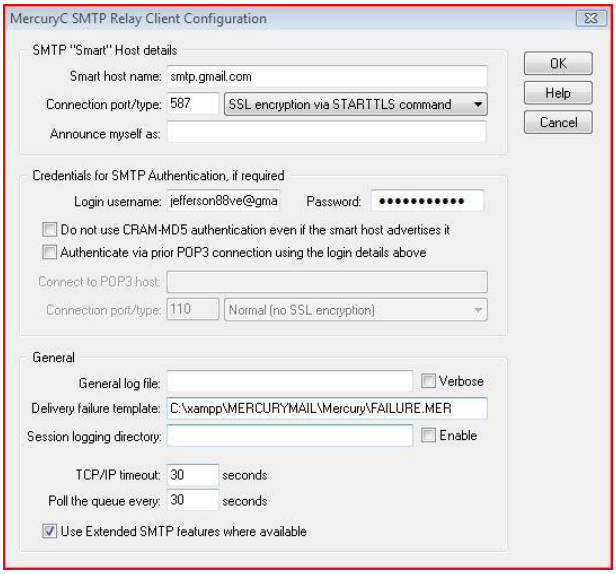


Figura 56. Configuración del cliente del SMTP del Mercury.

5.1.3. Configuración y Creación de Base de Datos en MySQL

Con el fin de mejorar la seguridad y controlar el acceso a las paginas administrativas del servidor y en especial a la base de datos, se siguen los siguientes pasos de configuracion:

En la Figura 57 se muestra como se ingresa desde el explorador web a la ruta localhost/security/index.php, donde se puede verificar el estado de la seguridad de cada uno de los componentes, identificando que todos se encuentran en color rojo, lo que significa que requieren atención para una adecuada configuración de seguridad.

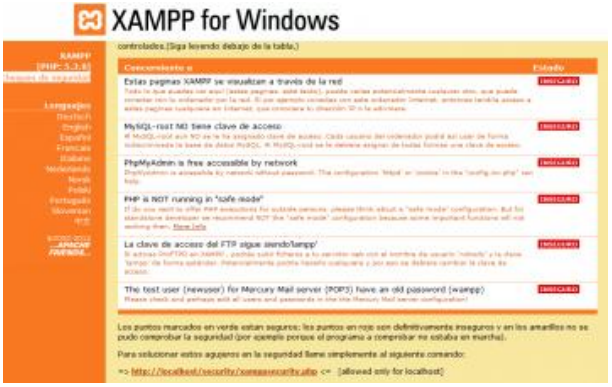


Figura 57. Ruta localhost/security/index.php en el navegador.

Luego de identificados los problemas de seguridad, se procede a agregar contraseñas de acceso al directorio de Xampp y a la consola de MySQL desde la ruta localhost/security/xamppsecurity.php en el explorador web como se visualiza en la Figura 58.

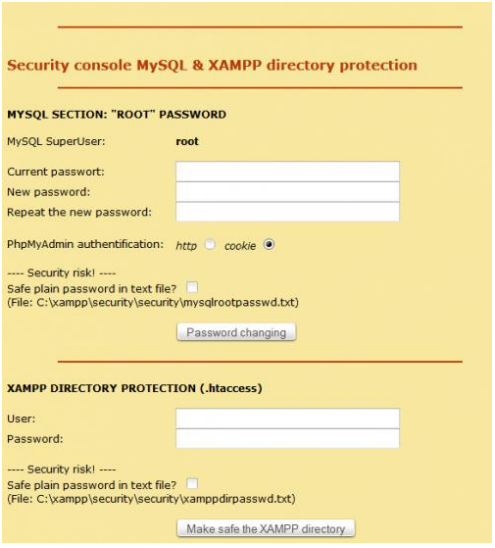


Figura 58. Ruta localhost/security/xamppsecurity.php.

Para la creación de la base de datos, primero se identifica el modelo relacional que se requiere para el manejo de la información de la plataforma web para el control del Observatorio Astronómico.

Luego de identificar el modelo y con ello las entidades (tablas), los atributos (campos), las llaves primarias y la relación de tablas, se desarrolla una base de datos en MySQL compuesta de 5 tablas como se aprecia en la Figura 59.

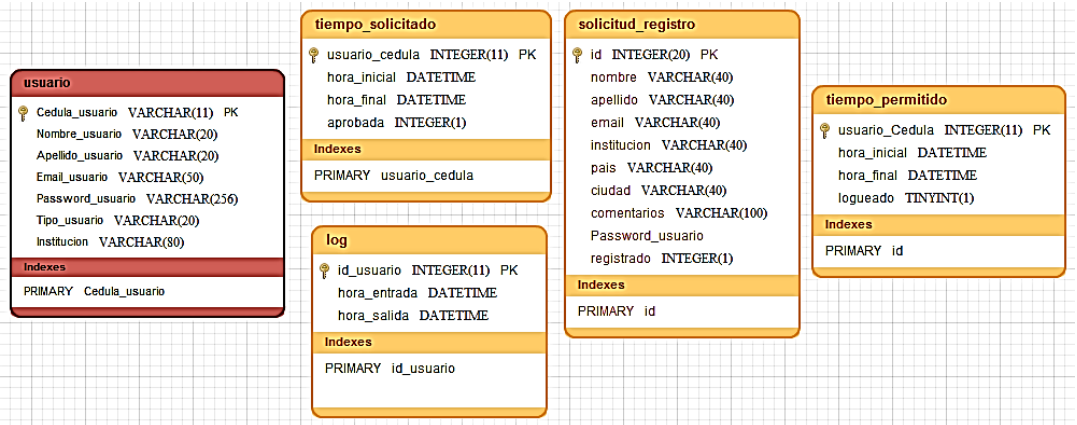


Figura 59. Composición de Base de Datos del control vía web del observatorio.

Para la creación de la base de datos se ejecuta la lógica expresada en el diagrama de flujo de la Figura 60, en lenguaje SQL [10] en el módulo MySQL de Xampp dando como resultado final la estructura del esquema "bd" en el gestor de base de datos de la Figura 61. El código completo para la creación de esta base de datos puede ser visualizado en el Anexo 3.

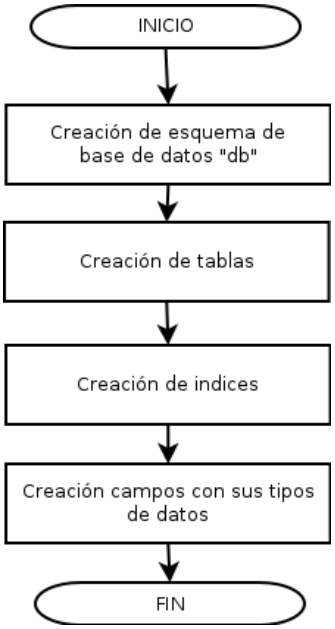


Figura 60. Diagrama de flujo para creación de base de datos.

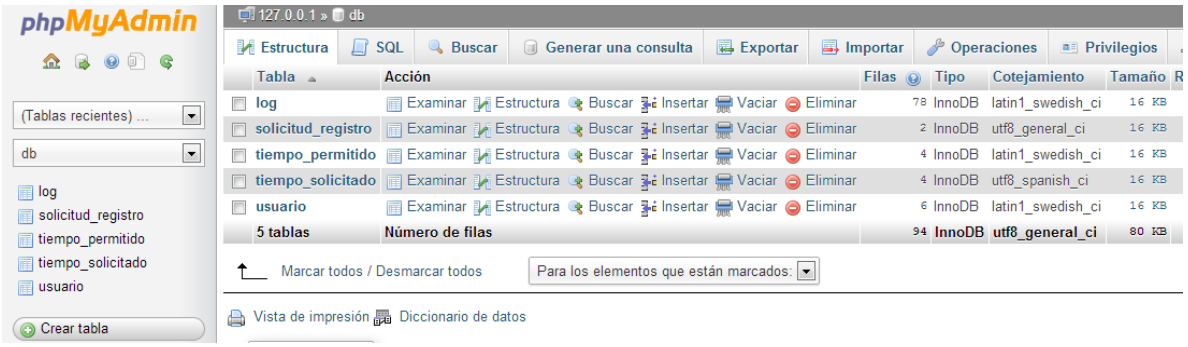


Figura 61. Base de datos “bd” creada en módulo MySQL de Xampp.

5.1.3.1. Descripción de las Tablas y Campos de la Base de Datos “bd”

Tabla Usuario: Almacena los datos de los usuarios y del administrador con su identificación respectiva. Los campos se listan de la siguiente manera:

Cedula_usuario: Guarda identificación personal de un usuario o administrador.

Nombre_usuario: Guarda el nombre de un usuario o administrador.

Apellido_usuario: Guarda el apellido de un usuario o administrador.

Email_usuario: Guarda el email de un usuario o administrador.

Password_usuario: Guarda la contraseña del usuario o administrador codificada.

Tipo_usuario: Guarda el tipo de cliente, en este caso si corresponde a administrador o usuario.

Institución: Guarda la institución a la que pertenece el usuario o administrador.

Tabla Tiempo_solicitado: Almacena información del horario solicitado por el usuario para usar el control del observatorio astronómico. Los campos se listan de la siguiente manera:

Usuario_cedula: Guarda identificación personal de un usuario solicitante.

Hora_inicial: Guarda la fecha y hora en que un usuario ha realizado una solicitud del control vía web del observatorio astronómico.

Hora_final: Guarda la fecha y hora en que un usuario ha decidido para finalizar.

Aprobada: Guarda un indicador entero donde si es uno corresponde a aprobada la solicitud y en caso contrario no es aprobaba.

Tabla Log: Almacena las horas de entrada y salida del usuario que ingresar al control del observatorio astronómico. Los campos se listan de la siguiente manera:

Id_Usuario: Guarda identificación personal de un usuario.

Hora_entrada: Guarda la fecha y hora en que un usuario ha ingresado al control vía web.

Hora_salida: Guarda la fecha y hora en que un usuario ha realizado una solicitud.

Tabla Solicitud_registro: Almacena la identificación del cliente que solicita permiso de registro a la plataforma. Los campos se listan de la siguiente manera:

Id: Guarda identificación personal de los potenciales usuario.

Nombre: Guarda el nombre del usuario.

Apellido: Guarda el apellido del usuario.

Email: Guarda el correo electrónico del usuario.

Institución: Guarda la institución a la que pertenece un usuario.

País: Guarda el país al que pertenece un usuario.

Ciudad: Guarda la ciudad al que pertenece un usuario.

Comentarios: Guarda el texto de observación de un usuario.

Password _usuario: Guarda la contraseña codificada de un usuario.

Registrado: Guarda un indicador entero con el que se identifica si el registro como usuario fue aprobado por el administrador de la plataforma.

Tabla Tiempo_permitido: Almacena los horarios permitidos del usuario para ingresar al control del observatorio astronómico. Los campos se listan de la siguiente manera:

Usuario_cedula: Guarda identificación personal de un cliente.

Hora_inicial: Guarda la fecha y hora en que un usuario se le ha aprobado el control vía web del Observatorio astronómico de la Universidad Tecnológica de Pereira.

Hora_final: Guarda la fecha y hora en que finaliza el tiempo permitido para el control vía web del Observatorio astronómico de la Universidad Tecnológica de Pereira.

Logueado: Guarda un indicador entero donde se identifica el estado de la sesión del usuario.

5.2. Plataforma Web

En esta sección del documento se describen las diferentes tecnologías, herramientas y procesos utilizados en el desarrollo del software necesario para el control vía web del Observatorio Astronómico de la Universidad Tecnológica de Pereira.

Para la construcción de la plataforma web se hizo uso de los siguientes lenguajes:

HTML: Es un lenguaje para desarrollar páginas web, el cual es conocido con el nombre de Hyper Text Markup Language [19], básicamente es un lenguaje de etiquetas o marcas que describen el contenido de un documento y se escribe en texto plano, las etiquetas de formato HTML se llaman elementos. Los elementos son parejas definidas de la siguiente forma y , donde el primero es el de inicio y el segundo es de cierre.

Los elementos se utilizan estrictamente dentro un inicio y fin. Una aplicación donde se hace uso de este lenguaje son los navegadores web, los cuales leen los documentos HTML y los muestra como páginas web, este no muestra explícitamente el lenguaje escrito sino que lo interpreta para determinar el contenido de la página web

Los atributos del lenguaje brindan información adicional con respecto a los elementos utilizados durante la elaboración de las páginas web, estos atributos son declarados en el encabezado de los elementos de apertura.

Los elementos más usados en la construcción de la plataforma web del observatorio han sido los formularios, tablas, listas, botones, entradas de texto, cuadros de división o divs, objects, embeds, imágenes y enlaces.

PHP: El lenguaje PHP hace las veces de procesamiento de la información enviada desde las páginas web, este se ejecuta en el lado del servidor pero provee gran velocidad de respuesta para páginas web. Su uso principal es en la conexión a base de datos y la ejecución de sentencias Sql, con el fin de interactuar con la página, pero con validación previa de datos. Además se proveen funcionalidades de conexiones de red y diferentes servidores como e-mail [20], para ello existen librerías que permiten las comunicaciones de otro tipo.

PHP permite usar funcionalidades comunes como fechas, manejo de arreglos, ciclos, condicionales, manejo de codificación para contraseñas, entre otros, que permiten desarrollar lógica de los datos que se reciben, de tal manera que se

puedan manipular y organizar para una forma más presentable y entendible, brindando el flujo de la información desde diferentes servicios hacia una interfaz común, conocida como página web.

JavaScript: Al inicio de JavaScript [21] en 1995, el principal objetivo fue controlar algunas entradas de validación que antiguamente habían sido del ordenador. Hoy en día JavaScript tiene una característica principal en el explorador Web dentro del mercado, no solo uniendo datos simples, JavaScript interactúa con la mayoría de los aspectos de la ventana del navegador y el contenido, además es un lenguaje de programación sumamente reconocido por su programación de objetos ya que realiza cálculos e interacciones complejas, incluyendo cierres y funciones, además de tener su propia implementación en Internet Explorer.

JavaScript se compone de tres partes: el núcleo (ECMAScript), el Document Object Model (DOM) y el modelo de objetos del navegador (BOM).

ECMAScript: Es un lenguaje definido en ECMA-262, el cual es instalado a los navegadores web. ECMA -262, es un estándar base que utilizan más lenguajes de programación.

The document object model (DOM): (DOM) El Document Object Model es una interfaz de programación de aplicaciones (API) para el XML, la cual se ha ampliado para uso de HTML. Los más conocidos mapas DOM son una página entera con una especie de jerarquía de nodos.

El modelo de objetos del navegador (BOM): Internet Explorer y Netscape Navigator 3.3 ofrecieron este objeto del navegador (BOM), el cual permite acceso y manipulación de la ventana del navegador. De esta manera los desarrolladores pueden interactuar con el navegador fuera del contexto de la página mostrada.

El lenguaje Javascript finalmente permite el dinamismo de las páginas web, de acuerdo a un esquema de programación orientada a eventos, es decir que cuando el usuario de un clic sobre un elemento, se tome una acción sobre el documento HTML como peticiones para llamar a archivos PHP en el que se procesen sentencias de conexión a base de datos y su respectiva manipulación, una herramienta que realiza dicha acción desarrollada en este lenguaje se conoce como Jquery [22].

jQuery: Es una librería que le proporciona al usuario una abstracción general aplicada a la capa web común, mediante secuencias de comandos.

Sus principales características permiten llevar a cabo las siguientes tareas:

- *Modelo del árbol (DOM) y localizar partes específicas de un documento de estructura en HTML.*

- *Responde a la interacción de un usuario, Eventos.*

- *Recuperar información de un servidor sin actualizar una página: este código patrón ha dado a conocer como Ajax, que originalmente significaba así como Asíncrono JavaScript y XML.*

SQL: Es un sistema que administra su motor de base de datos para visualizar el contenido de las tablas y sus relaciones para aplicaciones web. Las bases de datos almacenadas allí se conforman por un esquema de datos relacional el cual permite una forma lógica de modelar situaciones cotidianas y administrar sus datos dinámicamente. Los datos se almacenan en registros o filas de una tabla y cada fila se encuentra divida en diferentes columnas o campos, donde cada uno representa un valor que interpretado hace parte del mundo real. Cada tabla representa una relación entre objetos de un mundo real al cual se pueda extraer información adicional que la caracterice.

Estilos CSS: El lenguaje de una hoja de estilo es usado para describir el aspecto de una página web, su presentación al usuario y la manipulación en cascada por el navegador web. Cada campo de html se le puede aplicar una regla de estilos por medio del atributo style [23].

Una regla para la declaración y manejo de CCS están divididas en dos partes principales: un selector y una o más declaraciones, en donde el selector es normalmente el elemento en HTML que se desea modificar y cada declaración consta de una propiedad y un valor, la propiedad es el atributo de estilo que se desea cambiar, en donde un valor es asignado a cada propiedad.

El CCS permite a los desarrolladores especificar sus propios selectores conocidos con los nombres de atributos html id y class. El selector de id se utiliza para especificar un estilo para un elemento único y este se precede con un #, por otro lado el selector de class se utiliza para especificar un estilo a un conjunto de elementos y este se escribe tal cual como esta nombrado en la etiqueta html correspondiente.

5.2.1. Diagrama General

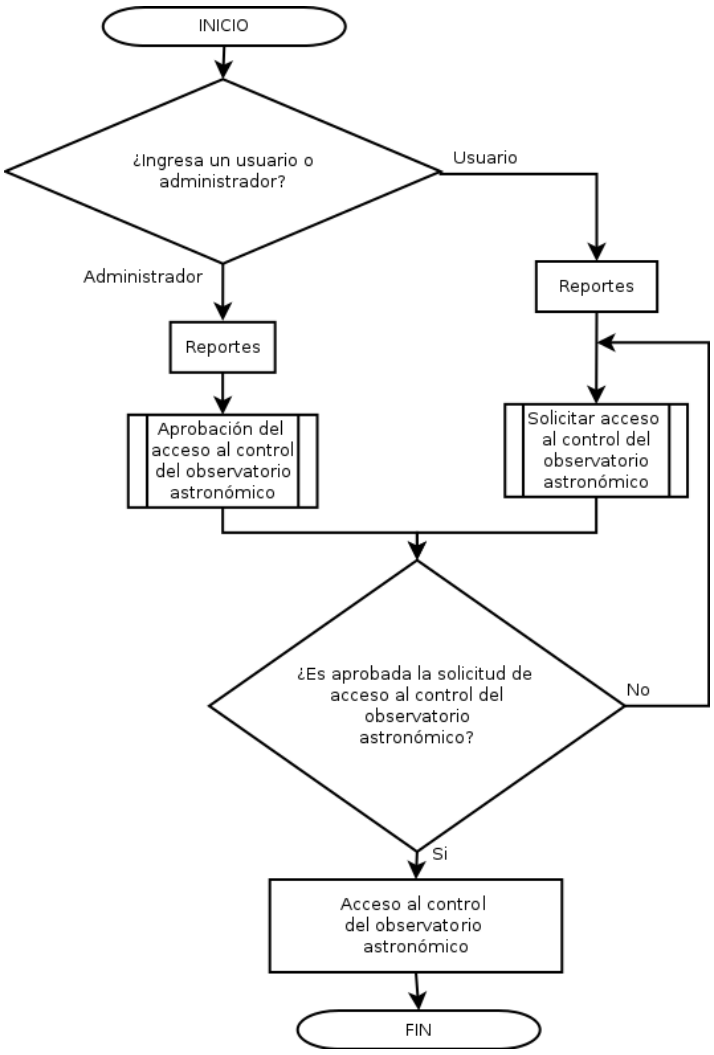


Figura 62. Diagrama de flujo general del aplicativo web para el control del observatorio Astronómico.

Como se puede visualizar en la Figura 62, la plataforma web está diseñada de modo que cualquier persona que desee ingresar a ella, primero debe de contar con los permisos y privilegios necesarios para lograr dicho acceso, Se encuentra diseñada además, teniendo en cuenta 2 tipos de usuario con roles diferentes dentro de la plataforma, uno de estos es el de rol de administrador, que es el encargado de habilitar el registro de nuevos usuarios dentro de la plataforma y de posteriormente permitir el acceso al control del Observatorio Astronómico de forma remota, a los usuarios registrados con el rol de usuario.

La plataforma cuenta ademas con diferentes reportes que muestran la informacion alojada en la base de datos, tanto para el rol de administrador como para el de usuario, permite entre otras cosas el cambio de datos de los usuarios registrados, verificar el reporte de actividad en la plataforma, verificar las solicitudes de registro pendientes y las solicitudes de acceso al control del observatorio.

5.2.2. Conexión a la Base de Datos

Mediante la conexión a la base de datos, se consigue establecer una fácil comunicación entre las páginas de la plataforma web y el gestor de base de datos MySQL, de forma que cada vez que se requiera insertar, consultar o actualizar un dato en particular de una o varias tablas, se pueda acceder y persistir sobre esta información con un simple llamado y el código SQL adecuado.

```
<?PHP
// Configuracion de la base de datos.
$dbhost = "localhost";      // Servidor
$dbuser = "root";           // Usuario
$dbpass = "xxxxxxx";        // Contraseña
$dbname = "db";             // Nombre de la base de datos

// Creando conexion.
$link = mysql_connect($dbhost,$dbuser,$dbpass); // Objeto de conexion a base de datos
        mysql_select_db($dbname,$link);         // Seleccionamos la base de datos
?>
```

Figura 63. Código principal de conexión hacia la base de datos.

En la Figura 63 se pueden apreciar los diferentes parámetros que deben ser tenidos en cuenta sobre la conexión hacia la base de datos, se define cual es el nombre del servidor, el nombre de usuario, la contraseña, y el nombre de la base de datos.

5.2.3. Notificación de Eventos Mediante Correo Electrónico

Dentro del funcionamiento de la plataforma web se hace uso del servidor de correo Electrónico Mercury, el cual fue tratado en el capítulo 5.2.2. Este servidor permite el envío automático de correos electrónicos con diferentes notificaciones remitidas hacia los usuarios o el administrador, En la Figura 64 se muestra un ejemplo de código donde se envía al administrador de la plataforma un correo notificándole sobre una persona que ha solicitado el registro en la misma, mediante un formulario que previamente ha llenado.

```
/*ENVIO DE E-MAIL A ADMINISTRADOR DE LA CUENTA*/
mail('CorreoObservatorio@gmail.com',"Solicitud de registro en Observatorio UTP",
" " . $nombre . " " .$apellido . " de la institución " . $institucion . "
ha solicitado el registro en la plataforma del Observatorio Astronómico de la UTP.
" . $pais . ": " . $ciudad . "");
```

Figura 64. Ejemplo de código de envío de Email.

Para el envío de estos correos se hace uso además de la función mail de PHP, esta función requiere los siguientes argumentos:

mail($destino, $asunto, $mensaje, $encabezados);

5.2.4. Ingreso a la Plataforma

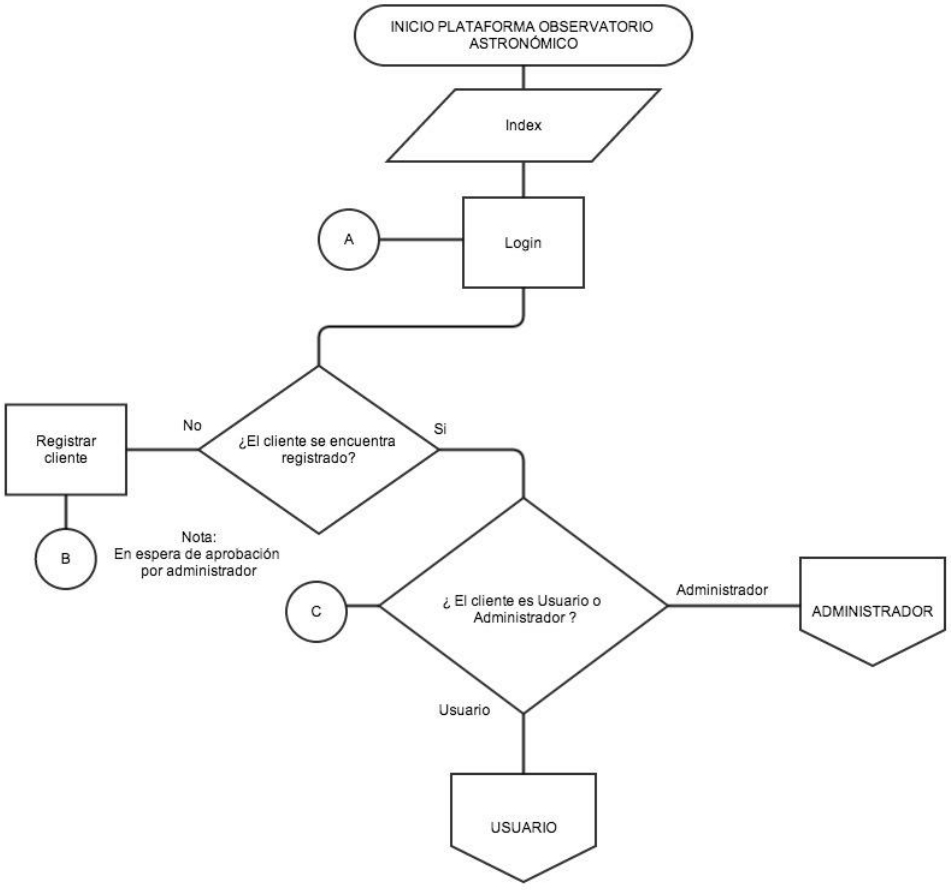


Figura 65. Diagrama de flujo para el ingreso a la plataforma web.

En la Figura 65 se describe el diagrama de flujo que muestra el proceso inicial con el que cuenta la plataforma para ingresar a ella. El ingreso a la plataforma se hace por medio de un usuario o un administrador.

De acuerdo a la Figura 66, al inicio de la plataforma se presentan dos campos, uno de usuario y otro de contraseña, y dos botones que permiten hacer el ingreso a la plataforma o su registro en caso de ser un nuevo usuario.

Figura 66. Pantalla de inicio de plataforma.

La pantalla de la Figura 66 se compone de dos entradas (input) y dos botones en HTML 5, los cuales contienen los valores de usuario y contraseña como se pueden visualizar en la Figura 67 y a los que posteriormente deben confirmarse para construir una sesión. Ver código completo en Anexo 4.

```
<p>Por favor ingrese codigo de usuario y contraseña</p>
<p align="center">      Usuario :     <input type="text" name="Id" id="Id">
                        Contraseña:   <input type="password" name="pass" id="pass">
</p>
<input type="button" name="button" id="button" value="Ingresar" onkeypress="redireccion()">
<a href="solicitud_registro.php" target="_self"><input type="button" name="boton" value="Registrarse"></a>
```

Figura 67. Aparte de código para la construcción de la página inicial.

El inicio en el sistema se diseñó de manera que al usuario solicitar el ingreso a la plataforma, se redirigirá hacia otra página que valida y corrobora la existencia del usuario sobre la plataforma, si no se encuentra registrado, se puede hacer uso del botón "Registrarse", donde se presenta una pantalla de registro en la plataforma.

A continuación se detallan sus procesos.

5.2.4.1. Autenticación y Manejo de Sesiones

```
    if (empty($user) || empty($pass)){
        echo json_encode(array('TEXT'=>'Debe digitar un usuario y contraseña valido','VALIDA'=>'1','USER'=>'No'));
        die("");
    }
```

Figura 68. Validación de datos de usuario registrado, datos completos.

Según la Figura 68, los datos tomados anteriormente (usuario y contraseña) se validan, evitando que se inserten datos erróneos o vacíos en la base de datos.

```
$checkuser      = mysql_query("SELECT Cedula_usuario FROM usuario WHERE Cedula_usuario='" . $user . "'");
$username_exist = mysql_num_rows($checkuser);
```

Figura 69. Validación de usuario ya registrado en el sistema.

Se consulta en la tabla "usuario" en la Figura 69 si existen los campos validados anteriormente, si los valores de los campos coinciden se logra la creación de la sesión.

```
if ($row['Tipo_usuario'] == 'ADMINISTRADOR') {
    $_SESSION["autentificado"]  = "SIA";
    echo json_encode(array('TEXT'=>'','VALIDA'=>'0','USER'=>'Ad'));
    die("");
}else if($row['Tipo_usuario'] == 'USUARIO') {
    $_SESSION["autentificado"]  = "SI";
    echo json_encode(array('TEXT'=>'','VALIDA'=>'0','USER'=>'Us'));
    die("");
}
```

Figura 70. Creación de sesión para usuario validado.

Como se puede apreciar en la Figura 70 la sesión se crea dependiendo del tipo de usuario, el cual puede ser administrador o usuario. Esta validación la realiza consultando en la base de datos el tipo de usuario de acuerdo al número de documento de identidad registrado.

5.2.4.2. Solicitud de Registro

La solicitud de registro en la plataforma se hace mediante el uso de un formulario que solicita en diferentes campos la información del usuario como se muestra en la Figura 71, ver código completo en Anexo 5.

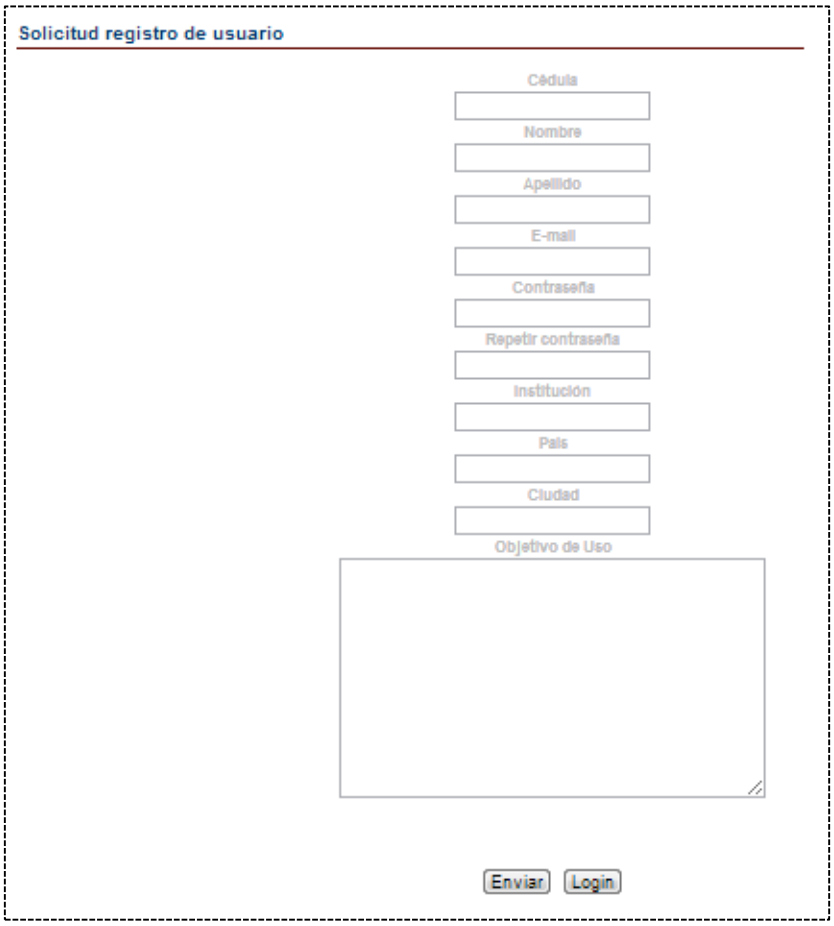


Figura 71. Formulario de solicitud de registro de usuario.

```
<div>
    <span class="show">Cédula</span><br>
    <INPUT TYPE="TEXT" NAME="id" id="cedula" class="texto">
</div>
<div>
    <span class="show">Nombre</span><br>
    <INPUT TYPE="TEXT" NAME="nombre" id="nombre" class="texto">
</div>
<div>
    <span class="show">Apellido</span><br>
    <INPUT TYPE="TEXT" NAME="apellido" id="apellido" class="texto">
</div>
<div>
    <span class="show">E-mail</span><br>
    <INPUT TYPE="TEXT" NAME="email" id="email" class="texto">
</div>
<div>
    <span class="show">Contraseña</span><br>
    <INPUT TYPE="password" id="pass" NAME="pass" class="texto">
</div>
<div>
    <span class="show">Repetir contraseña</span><br>
    <INPUT TYPE="password" id="pass2" NAME="pass2" class="texto">
</div>
<div>
    <span class="show">Institución</span><br>
    <INPUT TYPE="TEXT" id="institucion" NAME="institucion">
</div>
<div>
```

Figura 72. Formulario para la solicitud de registro en la plataforma.

El formulario de la Figura 72 se construye a partir de campos de entrada (input) con el fin de conservar los datos personales del usuario que solicita el registro.

```
if ($username_exist > 0){
    echo json_encode(array('TEXT'=> "Este número de identificacion ya se encuentra Registrado",'VALIDA'=>'1'));
    die ("");
}

if($_POST["pass"] != $_POST["pass2"]){
    echo json_encode(array('TEXT'=>"Las contraseñas no son iguales, por favor intentelo nuevamente",'VALIDA'=>'1'));
    die ("");
}
```

Figura 73. Validación de campos del formulario de registro de usuario.

Según la Figura 73, los datos son validados mediante sentencias realizadas en PHP con el fin de evitar errores en la creación del usuario, se verifica por ejemplo que las contraseñas coincidan en los dos campos donde se solicitan, o que el correo digitado tiene un formato adecuado para los estándares de Email.

```
$sql    = "INSERT INTO solicitud_registro(id,nombre,apellido,email,institucion,pais,ciudad,comentarios, Password_usuario)
            VALUES ('". $id ."','" .$nombre ."','" . $apellido ."','" . $email . "','" . $institucion . "', '" . $pais ."',
                    '" . $ciudad ."', '" . $comentarios . "', '" . $pass ."')";
mysql_query($sql) or die( json_encode(array('TEXT'=>"Ya se encuentra una solicitud de registro pendiente",'VALIDA'=>'1')) ); //
```

Figura 74. Inserción de la información del formulario de registro en la base de datos.

Después de validados los campos llenados por el usuario, en la Figura 74 estos son insertados sobre la tabla "solicitud registro" de la base de datos “bd”.

Para que el usuario puede ingresar a la plataforma, esta información debe ser validada por el administrador, lo cual es tratado en capítulo 5.2.6.1 de este documento.

5.2.5. Sesión de Usuario

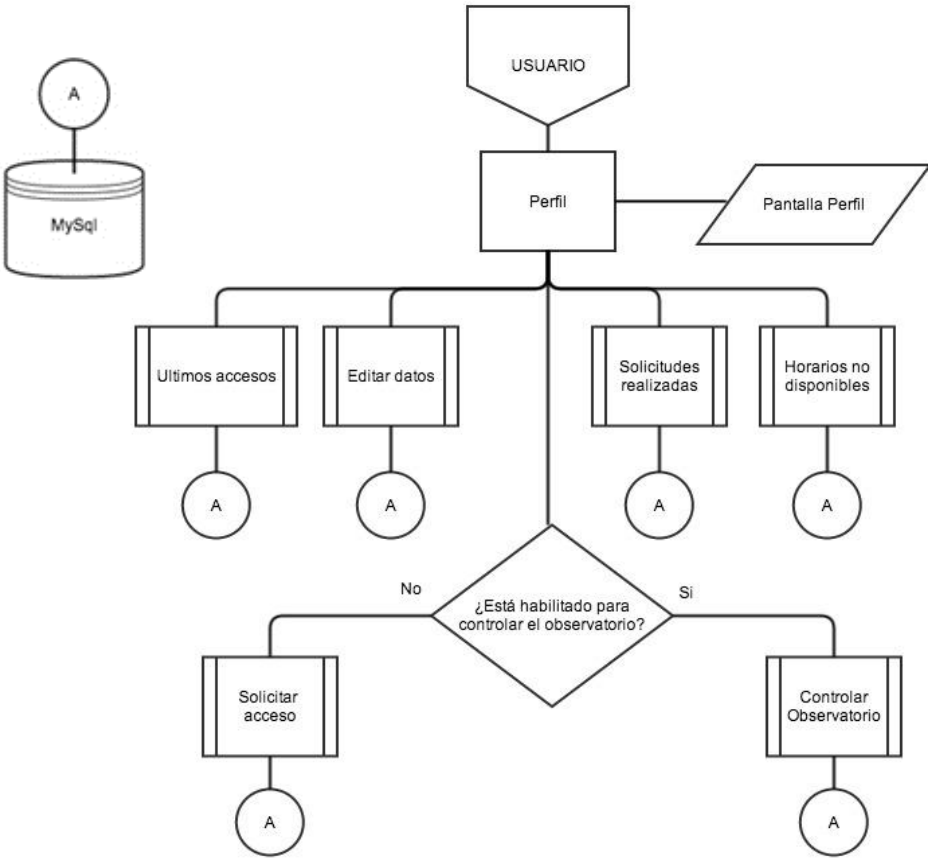


Figura 75. Diagrama de flujo de sesión de usuario.

El diagrama de flujo de la Figura 75, describe una página principal llamada "perfil", de la cual se desglosan diferentes subprocesos que en conjunto forman el perfil de un usuario sobre la plataforma web.

Cada sub proceso del diagrama de flujo corresponde a una pantalla en el aplicativo web, cada una está asociada con un código o script y generalmente al final de su ejecución realiza una o más transacciones a la base de datos “bd” sobre sus diferentes tablas.

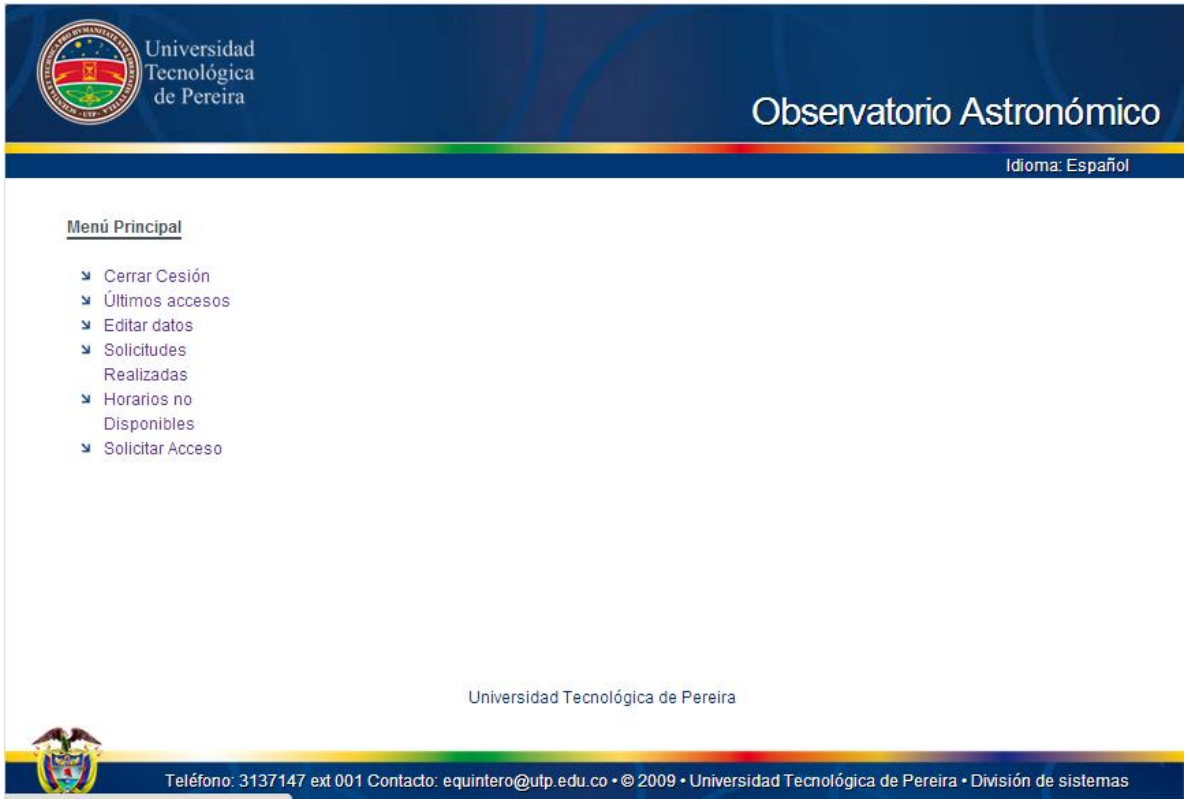


Figura 76. Página perfil con menú como lista html

La pantalla "perfil" de la Figura 76 carga un menú como una lista html, que tiene diferentes opciones para el perfil del usuario, al dar clic en estas opciones se visualizan formularios, tablas o reportes. El contenido de cada título del menú se recarga mediante la tecnología Ajax que se muestra en el código de la Figura 77, el cual hace parte de la librería Jquery y muestra como resultado la pantalla relacionada. Este proceso permite que el usuario solo deba recargar parte del contenido de toda la página en la que encuentra, optimizando la velocidad en la ejecución de este proceso. Ver código completo en Anexo 6.

```
$('a.ultimosAccesos').click(function(){
    $.ajax({
            data    :{ hrefs: '1'},
            type    :"POST",
            dataType:"html",
            async   :false,
            cache   :false,
            url     :'ultimos_accesos.php',
            success :function(data){$('div.content').html(data);},
            beforeSend:function(data){
                $('div.content').html('<img src="">');
                $('div.utp-menu-nav-subtitulo').html('<h2>Ultimos Accesos</h2>')
            },
            error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
        });
});
```

Figura 77. Trozo de código Ajax para cargar de forma independiente el contenido de cada opción del menú del usuario.

5.2.5.1. Reportes Disponibles Para los Usuario

Tres de las opciones del menú corresponden a los reportes necesarios para el perfil del usuario.

Últimos Accesos:

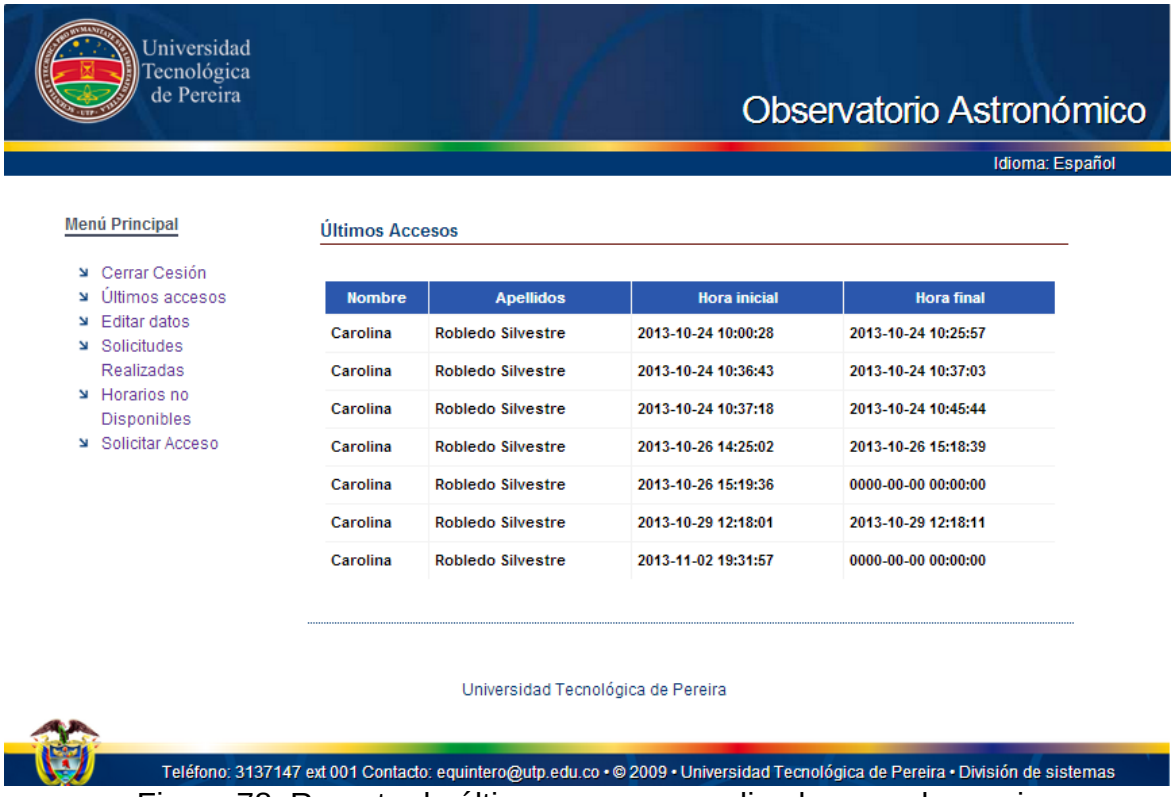


Figura 78. Reporte de últimos accesos realizados por el usuario.

La pantalla de la Figura 78 muestra una tabla con encabezados titulados como “Nombre”, ”Apellidos”, ”Hora Inicial” y ”Hora final”, los cuales detallan los rangos de horario con los últimos ingresos a la plataforma. Ver código completo para la generación de este reporte en Anexo 7.

```
include('config.php');
$user       = $_SESSION['Id'] ;
$result     = mysql_query("SELECT * FROM usuario u LEFT JOIN log t on (u.cedula_Usuario=t.id_usuario)
              WHERE NOT tipo_usuario ='ADMINISTRADOR' AND Cedula_usuario=$user ORDER BY Apellido_usuario,Nombre_usuario");
```

Figura 79. Consulta a las tablas usuario y log, sobre los accesos a la plataforma.

En la Figura 79 se consulta a la tabla “usuario” unida a la tabla “log” de la base de datos; los campos “Nombre usuario”, “Apellido usuario”, “hora inicial” y “hora final”

donde existan registros diferentes al tipo de usuario administrador y coincida con el número de documento del usuario que realiza la consulta.

```
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1" frame="void" bordercolor="#F3F3F3">
   <thead class="style3">
       <tr>
           <td bgcolor="#2b57ad" align="center"><span class="show">Nombre</span></td>
           <td bgcolor="#2b57ad" align="center"><span class="show">Apellido</span></td>
           <td bgcolor="#2b57ad" align="center"><span class="show">Hora inicial</span></td>
           <td bgcolor="#2b57ad" align="center"><span class="show">Hora final</span></td>
       </tr>
   </thead>
```

Figura 80. Código para creación de tabla donde se muestra el contenido de la consulta.

Se crea el contenido de una tabla html con sus respectivos encabezados mostrados en la Figura 80 y su cuerpo el cual se visualiza en la Figura 81, este se genera dinámicamente recorriendo los registros devueltos por la consulta y ubicando los datos en el orden de cada campo de la misma.

```
  <tbody>
      <?php
          /* -- Mostramos los registros -- */
          while ($row = @mysql_fetch_array($result)) {
              echo '<tr class="style6">
                      <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["Nombre_usuario"] . '</span></td>
                      <td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] . '</span></td>
                      <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_entrada"] . '</span></td>
                      <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_salida"] . '</span></td>
                    </tr>';
          }
          @mysql_free_result($result);
      ?>
  </tbody>
```

Figura 81. Consulta a la base de datos e impresión de datos sobre la tabla creada.

Solicitudes de Acceso:

Figura 82. Reporte con solicitudes realizadas para el control vía web del observatorio.

De igual forma en la pantalla de la Figura 82 se muestra una tabla con encabezados titulados como; “Nombre”, “Apellidos”, “Hora inicial”, “Hora final” y “Estado”, este muestra al usuario todas las solicitudes realizadas en diferentes rangos de horario, para acceder al control del Observatorio Astronómico de la Universidad Tecnológica de Pereira, las cuales se aprueban o desaprueban por el administrador. (Tema tratado en el capítulo 5.2.6.1.) El código relacionado con el reporte de solicitudes de acceso, puede ser visto en el Anexo 8.

```php
<?php
    include('config.php');
    $user   = $_SESSION['Id'] ;
    $result = mysql_query("SELECT * FROM usuario u LEFT JOIN tiempo_solicitado t ON (u.cedula_Usuario=t.usuario_cedula)
                    WHERE Cedula_usuario=$user ORDER BY Apellido_usuario,Nombre_usuario");
?>
```

Figura 83. Consulta a la base de datos “bd” de los horarios asignados al usuario para el acceso al control del Observatorio.

Para lograr realizar este reporte, se muestra en la Figura 83 cómo se consulta a la tabla “usuario” unida a la tabla “tiempo solicitado” de la base de datos los campos; “Nombre usuario”, “Apellido usuario”, “hora inicial”, “hora final” y “aprobado”, de modo que devuelva los datos necesarios para reconocer que franjas horarias se encuentran disponibles para el acceso al control del observatorio.se consulta a la tabla “usuario” unida a la tabla “tiempo solicitado” de la base de datos los campos; “Nombre usuario”, “Apellido usuario”, “hora inicial”, “hora final” y “aprobado”, de modo que devuelva los datos necesarios para reconocer que franjas horarias se encuentran disponibles para el acceso al control del observatorio.

```php
<?php

/* -- Mostramos los registros --*/
while ($row = @mysql_fetch_array($result)) {
    echo '<tr class="style6">
            <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["Nombre_usuario"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_inicial"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_final"] . '</span></td>';
            if($row["aprobada"] == '0'){
                echo '<td  bgcolor="#FFFFFF"><span class="show">Pendiente</span></td></tr>';
            }else{
                echo '<td  bgcolor="#FFFFFF"><span class="show">Aprobado</span></td></tr>';
            }

}
@mysql_free_result($result);
?>
```

Figura 84. Código que devuelve la consulta de la base de datos hacia el reporte solicitado en una tabla.

Se crea el contenido de una tabla html con su cuerpo y sus respectivos encabezados como se puede visualizar en la Figura 84, esta se construye dinámicamente recorriendo los registros traídos por la consulta y se organizan según la forma en cómo son seleccionados.

Horarios no disponibles para el Acceso:

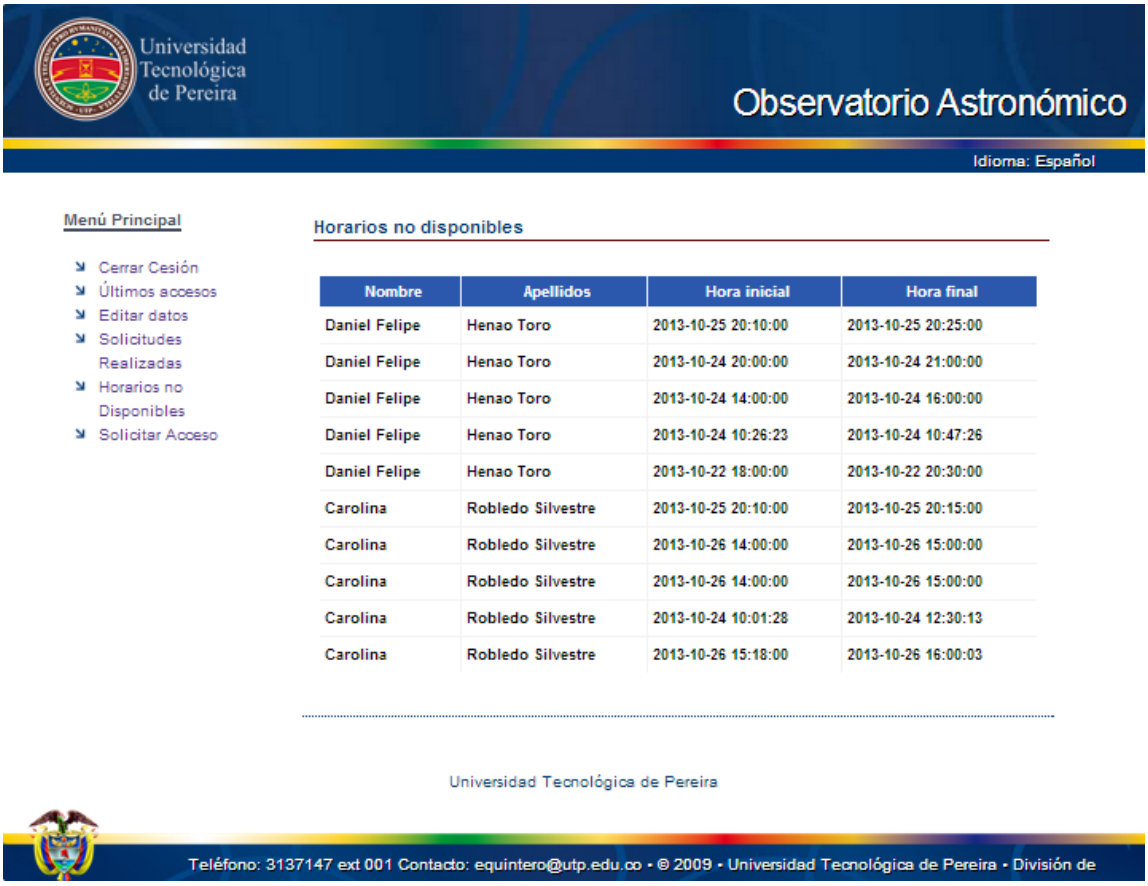


Figura 85. Reporte de Horarios utilizados por otros Usuarios.

De la misma forma la pantalla de la Figura 85 se presenta una tabla con encabezados titulados como; “Nombre”, “Apellidos”, “Hora inicial” y “Hora final”, quienes detallan que rangos horarios del acceso al control del Observatorio Astronómico de la Universidad Tecnológica de Pereira están ocupados por otros usuarios. Para ver el código completo que permite la generación de este reporte, remítase al Anexo 9.

```php
<?php
    include('config.php');
    $user       = $_SESSION['Id'] ;
    $result     = mysql_query("select * from usuario u left join tiempo_solicitado t on (u.cedula_Usuario=t.usuario_cedula)
                        where Cedula_usuario!='' and hora_inicial order by Apellido_usuario,Nombre_usuario");
?>
```

Figura 86. Consulta a la base de datos para la generación del reporte de tiempos ya asignados a otros usuarios.

Según la Figura 86, se consulta a la tabla “usuario” unida a la tabla “tiempo solicitado” de la base de datos; los campos “Nombre usuario”, “Apellido usuario”, “hora inicial” y “hora final” de todos los usuarios.

```
    <tbody>
<?php
    /* -- Mostramos los registros -- */
    while ($row = @mysql_fetch_array($result)) {
                echo '<tr class="style6">
                            <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["Nombre_usuario"] . '</span></td>
                            <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["Apellido_usuario"] . '</span></td>
                            <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["hora_inicial"] . '</span></td>
                            <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["hora_final"] . '</span></td>
                        </tr>';
    }
    @mysql_free_result($result);
?>
    </tbody>
```

Figura 87. Código que devuelve la consulta de la base de datos hacia el reporte de horarios no disponibles.

Finalmente se crea el contenido de la tabla html con cada encabezado y su cuerpo visualizado en la Figura 87, el cual es creado dinámicamente recorriendo los resultados de los registros de la consulta y se organizan según la forma como se encuentren los campos.

5.2.5.2. Edición de datos Personales

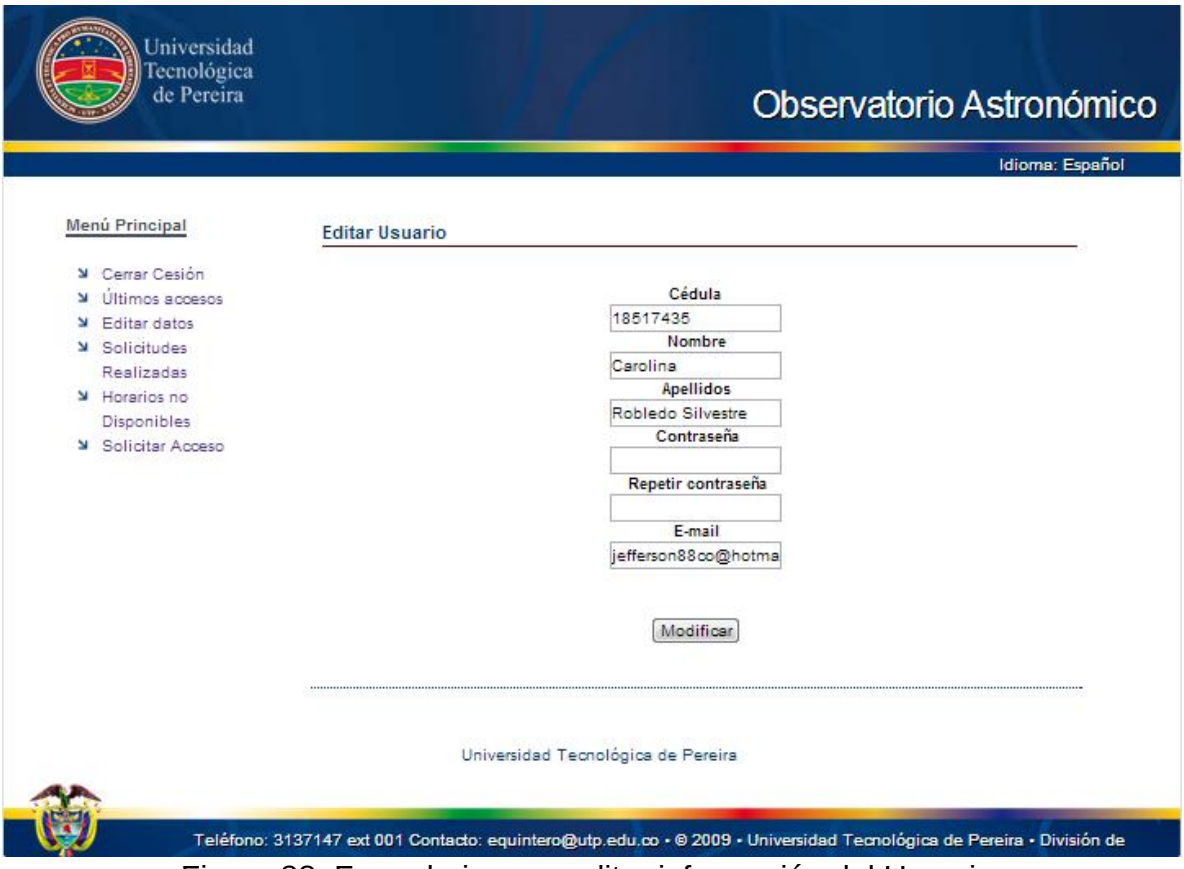


Figura 88. Formulario para editar información del Usuario.

La pantalla de la Figura 88 presenta un formulario que contiene la información personal del usuario, la cual está registrada en el sistema, de tal forma que se pueda modificar. Para visualizar el código completo que permite la edición de datos personales remítase al Anexo 10.

```php
$id         = $_SESSION["Id"];
//Creamos la sentencia SQL y la ejecutamos
$sSQL       = "SELECT * FROM usuario WHERE Cedula_usuario='$id'";
$result     = mysql_query($sSQL);
$row        = mysql_fetch_array($result);
$cedula     = $row["Cedula_usuario"];
$nombre     = $row["Nombre_usuario"];
$apellido   = $row["Apellido_usuario"];
$email      = $row["Email_usuario"];
$pass       = $row["Password_usuario"];
```

Figura 89. Código para la generación del formulario de edición de datos personales.

Como se puede apreciar en la Figura 89, los datos de la tabla "usuario" del usuario en sesión se consultan para asignarse en diferentes variables de php e incluirlas sobre los campos de entrada de texto (input) html pertenecientes a este formulario, teniendo en cuenta que el campo asociado a la cédula es de solo lectura, no permite ser modificado.

```php
    <div>
        <span class="show">Cédula </span><br>
            <INPUT TYPE="TEXT" NAME="id" id="cedula" value="' . $cedula . '" class="texto">
    </div>
    <div>
        <span class="show">Nombre</span><br>
            <INPUT TYPE="TEXT" NAME="nombre" id="nombre" value="' . $nombre . '" class="texto">
    </div>
    <div>
        <span class="show">Apellido</span><br>
            <INPUT TYPE="TEXT" NAME="apellido" id="apellido" value="' . $apellido . '" class="texto">
    </div>
    <div>
        <span class="show">Contraseña</span><br>
            <INPUT TYPE="password" id="pass" NAME="pass" class="texto">
    </div>
    <div>
        <span class="show">Repetir contraseña</span><br>
            <INPUT TYPE="password" id="pass2" NAME="pass2" class="texto">
    </div>
    <div>
        <span class="show">E-mail</span><br>
            <INPUT TYPE="TEXT" NAME="email" id="email" value="' . $email . '" class="texto">
    </div>
    <br>
            <INPUT TYPE="button" id="Modificar" value="Modificar">
```

Figura 90. Formulario para la edición de datos personales del usuario.

Al dar clic en el botón "modificar" de la Figura 90 se procesa una petición por medio de la tecnología Ajax mostrada en la Figura 91, si esta es exitosa indica que la modificación de los datos personales han sido actualizados.

```
$('#Modificar').click(function(){
    $.ajax({
                data    :{  cedula      : $('#cedula').val(),
                            nombre      : $('#nombre').val(),
                            apellido    : $('#apellido').val(),
                            pass        : $('#pass').val(),
                            pass2       : $('#pass2').val(),
                            email       : $('#email').val(),
                         },
                type    :"POST",
                dataType:"html",
                async   :false,
                cache   :false,
                url     :'xxxxx.php',
                success :function(data){ ....
```

Figura 91. Código Ajax para actualizar los datos personales del usuario.

Mientras se procesa la petición, se validan los datos como se muestra en la Figura 92 para evitar realizar actualizaciones erróneas sobre la base de datos. En caso de no cumplir con las condiciones de validación necesarias; como por ejemplo una contraseña vacía, entonces se informa al usuario mediante un cuadro de dialogo el mensaje “las contraseñas no son iguales, por favor inténtelo nuevamente”.

```
if ($_POST['pass'] != $_POST['pass2']) {
    echo "Las contraseñas no son iguales, por favor intentelo nuevamente";
    die("");
}
```

Figura 92. Código para validación de contraseña.

Al cumplir con las condiciones de validación preestablecidas según la Figura 93, se realiza la actualización en la tabla "usuario" con los datos recibidos de la petición Ajax como "e-mail", "nombre", "apellidos" y "contraseña"."e-mail", "nombre", "apellidos" y "contraseña".

```
/* -- Creamos la sentencia SQL y la ejecutamos - UPDATE -- */
$sSQL = "UPDATE usuario SET Nombre_usuario ='$nombre',Apellido_usuario ='$apellido',
        Email_usuario ='$email',Password_usuario ='$pass' WHERE Cedula_usuario ='$cedula'";
mysql_query($sSQL) or die("Error en la creacion. MySQL dice:" . mysql_error());
```

Figura 93. Sentencia SQL para la actualización de los datos del usuario en la base de datos “bd”.

5.2.5.3. Solicitud de Acceso para Control Web del Observatorio

Figura 94. Formulario de solicitud de acceso a control de la plataforma vía web.

La pantalla de la Figura 94 permite la selección de un rango horario para solicitar el acceso al control del Observatorio Astronómico de la Universidad Tecnológica de Pereira, el cual debe ser aprobado por el administrador posteriormente (tema que es tratado en el capítulo 5.2.6.1). Los códigos completos para la solicitud de acceso al control web del Observatorio, pueden ser visualizados en el Anexo 11.

```
/* -- Creamos la sentencia SQL y la ejecutamos -- */
    $sSQL      = "SELECT Cedula_usuario,Nombre_usuario,Apellido_usuario  FROM usuario WHERE Cedula_usuario='$id'";
    $result    = mysql_query($sSQL);
```

Figura 95. Consulta SQL a la base de datos.

En la Figura 95 se consulta en la tabla "usuario" para hacer uso de los campos "Nombre usuario", "Cedula usuario" de la base de datos y recorrer dichos registros e insertarlos en un combo html o select.

```
echo '<option value=' . $row["Cedula_usuario"] . '>' . $row["Nombre_usuario"] . ' ' . $row["Apellido_usuario"] . '</option>';
```

Figura 96. Código que devuelve el nombre de usuario que realiza la solicitud.

Los datos "Nombre usuario" y "Cedula usuario" corresponde al valor y el identificador del select respectivamente según la Figura 96.

```
var show = $('#1').datetimepicker({
        dateFormat: "yy'-'mm'-'dd",
        separator: ' ',
        ampm: false,
        onSelect:function(data){
            $('#1').attr('value',data);
        }
});
```

Figura 97. Código para la inserción del formato de fecha mediante un pluggin de Jquery.

Por medio del código de la librería Jquery de la Figura 97, se carga un pluggin de selección de fecha y hora sobre dos campos de entrada de texto (input); el primero según la pantalla de la Figura 98 corresponde a la fecha y hora de inicio y el segundo a la fecha y hora fin.

Figura 98. Pluggin de selección de fecha y hora.

```
$('#solicitar').click(function(){
    $.ajax({
                data    :{ cedula: $('#cedula').val(),
                           t1:  $('#1').val(),
                           t2:  $('#2').val()
                         },
                type    :"POST",
                dataType:"html",
                async   :false,
                cache   :false,
                url     :'xxxxxx.php',
                success :function(data){
                $('#dialog').html('<p>' + data + '</p>');
                    $( "#dialog" ).dialog({
                        autoOpen: true,
                        modal:true,
                        show: {
                            effect: "slide",
                            duration: 1000
                        },
                        hide: {
                            effect: "fold",
                            duration: 1000
                        }
                    });
                },
                error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
    });
});
```

Figura 99. Código Ajax para insertar el rango horario seleccionado por el usuario

Al dar clic en el botón "solicitar", según la Figura 99 se procesa una petición por medio de la librería Ajax de Jquery, si esta petición es exitosa se indica que la solicitud del usuario se ha registrado sin problemas.

```
$result     = mysql_query("SELECT count(*) c FROM tiempo_permitido WHERE (hora_inicial between '$t1' and '$t2')
                           OR (hora_final between '$t1' and '$t2')") or die("Couldn't query the user-database.");
$num        = mysql_fetch_array($result);
```

Figura 100. Consulta SQL a la base de datos “bd” para verificar el cruce de horarios con otro ya asignado.

Al recibir la hora inicio y fin de la petición Ajax, de acuerdo a la Figura 100 se consulta la cantidad de datos existentes en la tabla "tiempo permitido", los cuales se deben obtener cuando los campos; hora inicial y final, estén entre la hora inicial y final solicitadas, con el fin de validar si existe un cruce de horarios.

```
$sql = "INSERT INTO tiempo_solicitado(usuario_Cedula,hora_inicial,hora_final) VALUES ($cedula,'$t1','$t2')";
mysql_query($sql) or die("");
mail('xxxxx@gmail.com',"Solicitud de ingreso a Observatorio UTP",
       "El señor(a), $nombre $apellido ha solicitado el ingreso al Observatorio Astronómico de la UTP desde $t1 hasta $t2");
```

Figura 101. Inserción de datos en tabla tiempo_solicitado de la base de datos “bd”.

Luego se procede a validar los datos de rango de horario seleccionados por el usuario y de acuerdo con la Figura 101 se inserta en la tabla "tiempo solicitado" de la base de datos “bd”, dichos rangos. De tal manera que puedan ser mostrados en

el reporte "solicitudes pendientes" del administrador, tema propuesto en el capítulo 5.2.6.1.

5.2.6. Sesión de Administrador

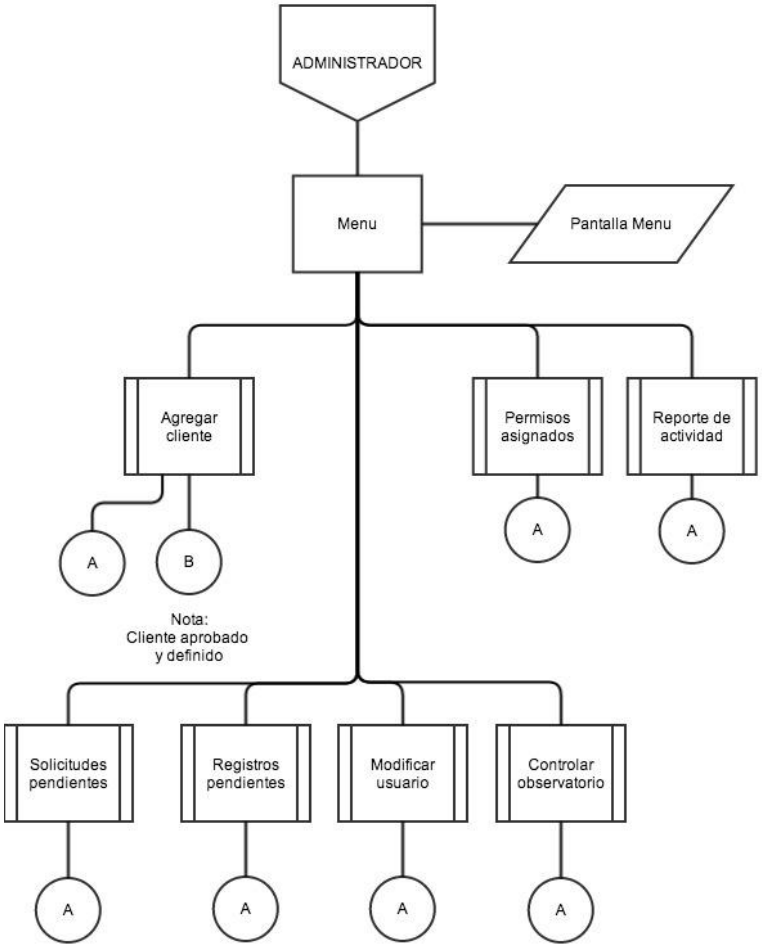


Figura 102. Diagrama de flujo de sesión de administrador.

El diagrama de flujo de la Figura 102 describe inicialmente una página principal llamada "menu", de la cual se despliegan diferentes subprocesos que en forma conjunta integran el perfil de administrador sobre la plataforma web, en la cual se realizan operaciones de control de usuario.

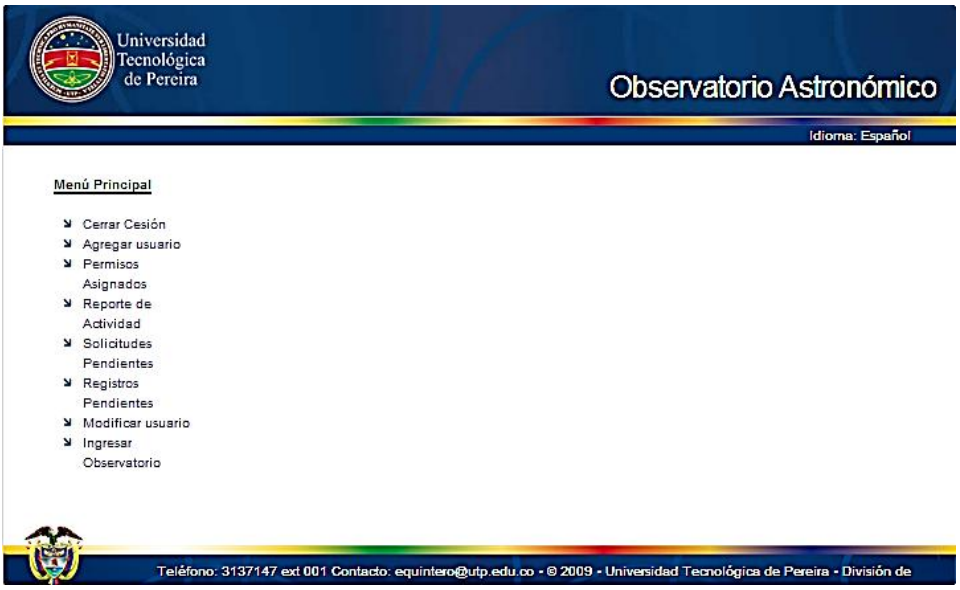


Figura 103. Menú principal de administrador.

La pantalla de la Figura 103 muestra el contenido del "menú" que consiste en una lista html, el cual tiene diferentes opciones para el perfil de un administrador, al dar clic en estas opciones se visualizan formularios y reportes. El contenido de cada título del menú se recarga mediante la tecnología Ajax que hace parte de la librería Jquery y muestra como resultado la pantalla relacionada.

5.2.6.1. Reportes Disponibles Para el Administrador.

Los reportes del administrador corresponden a cuatro de las opciones del menú:

Permisos Asignados:

Figura 104. Reporte de permisos asignados por el administrador.

La pantalla de la Figura 104 muestra una tabla con encabezados titulados como; “Cédula”, “Nombre”, ”Apellidos”, ”Hora Inicial” y ”Hora final”, los cuales detallan los rangos de horario asignados a cada usuario para el control del Observatorio Astronómico de la Universidad Tecnológica de Pereira. El código completo que describe el reporte "permisos asignados" se encuentra en el Anexo 12.

```
$result = mysql_query("SELECT * FROM usuario u LEFT JOIN tiempo_permitido t ON (u.cedula_Usuario=t.usuario_Cedula)
                       WHERE NOT tipo_usuario ='ADMINISTRADOR' AND hora_inicial!=''
                       ORDER BY hora_inicial DESC,hora_final DESC,Apellido_usuario,Nombre_usuario LIMIT 20");
```

Figura 105. Consulta SQL a la base de datos “bd” de todos los rangos horarios permitidos de cada usuario.

Para mostrar los permisos asignados, en la Figura 105 se consultan todos los campos de la tabla “usuario” unida con la tabla “tiempo permitido” de la base de datos, de modo que el tipo de usuario no sea administrador.

```
<tbody>
<?php
    /* -- Se muestra los registros --*/
    while ($row = @mysql_fetch_array($result)) {

        echo '<tr class="style6">
                <td bgcolor="#FFFFFF" height="20"><span class="show">' . $row["Cedula_usuario"] . '</span></td>
                <td bgcolor="#FFFFFF"><span class="show">' . $row["Nombre_usuario"] . '</span></td>
                <td bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] . '</span></td>
                <td bgcolor="#FFFFFF"><span class="show">' . $row["hora_inicial"] . '</span></td>
                <td bgcolor="#FFFFFF"><span class="show">' . $row["hora_final"] . '</span></td>
            </tr>';
    }
    @mysql_free_result($result);
?>
</tbody>
```

Figura 106. Tabla HTML correspondiente al reporte de permisos asignados.

Luego de hacer la consulta en la base de datos, se recorren los datos obtenidos para ingresarlos en cada columna perteneciente al cuerpo de la tabla del reporte permisos asignados como se puede observar en la Figura 106.

Reporte de Actividad:

Últimos Accesos

Cedula	Nombre	Apellidos	Hora inicial	Hora final
18517435	Carolina	Robledo Silvestre	2013-11-02 19:31:57	2013-11-02 19:59:38
18517435	Carolina	Robledo Silvestre	2013-10-29 12:18:01	2013-10-29 12:18:11
18517434	Jefferson	Martinez	2013-10-26 15:18:48	2013-10-26 15:19:17
18517435	Carolina	Robledo Silvestre	2013-10-26 14:25:02	2013-10-26 15:18:39
18517434	Jefferson	Martinez	2013-10-26 14:24:12	2013-10-26 14:24:44
18517434	Jefferson	Martinez	2013-10-24 19:19:47	2013-10-24 19:36:53
1088273554	Daniel Felipe	Henao Toro	2013-10-24 17:55:39	2013-10-24 17:55:53
18517434	Jefferson	Martinez	2013-10-24 14:28:17	2013-10-24 14:29:15

Figura 107. Reporte de actividad de últimos accesos a la plataforma.

En la pantalla de la Figura 107 se muestra una tabla con encabezados titulados como; “Cédula”, “Nombre”, ”Apellidos”, ”Hora Inicial” y ”Hora final”, los cuales detallan los rangos de horario de los últimos ingresos a la plataforma de cada usuario registrado en la plataforma. El código completo de reporte de actividad puede ser visualizado en Anexo 13.

```
$result = mysql_query("SELECT * FROM usuario u LEFT JOIN log t ON (u.cedula_Usuario=t.id_usuario)
                       WHERE tipo_usuario ='ADMINISTRADOR' AND hora_salida!='0000-00-00 00:00:00'
                       OR tipo_usuario='USUARIO' AND hora_salida!='0000-00-00 00:00:00'
                       ORDER BY hora_entrada DESC,hora_salida DESC,Apellido_usuario,Nombre_usuario LIMIT 20");
```

Figura 108. Consulta SQL a la base de datos “bd” sobre la actividad de usuarios en el control del observatorio astronómico.

Para mostrar la información de reporte de actividad, se realiza una consulta de todos los campos de la tabla “usuario” unida con la tabla “log” según se muestra en la Figura 108; los cuales se obtienen de acuerdo a dos condiciones, primero que el tipo de usuario sea cualquiera, y segundo que las horas de entrada y salida sean diferentes a la cadena '0000-00-00 00:00:00'.

```
/* -- Mostramos los registros -- */
while ($row = @mysql_fetch_array($result)) {
    echo '<tr class="style6">
            <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["Cedula_usuario"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["Nombre_usuario"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_entrada"] . '</span></td>
            <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_salida"] . '</span></td>
        </tr>';
}
@mysql_free_result($result);
```

Figura 109. Tabla HTML del reporte de actividad.

Luego de realizar la consulta se recorre el resultado obtenido para ingresarlos en cada columna del reporte, el cual también posee un encabezado y un cuerpo que se pueden visualizar en la Figura 109.

Solicitudes Pendientes de Acceso a Control Web del Observatorio:

Solicitudes Pendientes

Cédula	Nombre	Apellidos	Hora inicial	Hora final	Estado	Asignar
1088273554	Daniel Felipe	Henao Toro	2013-10-25 20:10:00	2013-10-25 20:25:00	Pendiente	Aprobar Rechazar
18517435	Carolina	Robledo Silvestre	2013-10-25 20:10:00	2013-10-25 20:15:00	Pendiente	Aprobar Rechazar
18517435	Carolina	Robledo Silvestre	2013-10-26 14:00:00	2013-10-26 15:00:00	Pendiente	Aprobar Rechazar
18517435	Carolina	Robledo Silvestre	2013-10-26 14:00:00	2013-10-26 15:00:00	Pendiente	Aprobar Rechazar

Figura 110. Reporte de solicitudes pendientes para el acceso al control vía web del observatorio.

La pantalla de la Figura 110 en su contenido muestra una tabla con una funcionalidad adicional, consiste en aprobar o desaprobar las solicitudes enviadas por cada usuario (capítulo 5.2.5.3), la tabla tiene encabezados titulados como; “Cédula”, “Nombre”, ”Apellidos”, ”Hora Inicial”, ”Hora final”, “Estado” y “Asignar”, los cuales detallan los rangos horarios solicitados por los usuarios de la plataforma web. En el reporte de solicitudes pendientes el código construye toda la presentación inicial más la acción relacionada con la aprobación o desaprobación de ellas, El código completo puede ser visualizado en el Anexo 14.

```
$user   = $_SESSION['Id'];
$result = mysql_query("SELECT * FROM usuario u LEFT JOIN tiempo_solicitado t ON (u.cedula_Usuario=t.usuario_cedula)
                       WHERE Tipo_usuario='USUARIO' ORDER BY Apellido_usuario,Nombre_usuario");
```

Figura 111. Consulta a la base de datos “bd”, sobre las solicitudes de acceso pendientes por aprobación.

Para mostrar la información de solicitudes pendientes se ejecuta la consulta de la Figura 110 de todos los campos de la tabla “usuario” unida con la tabla “tiempo solicitado".

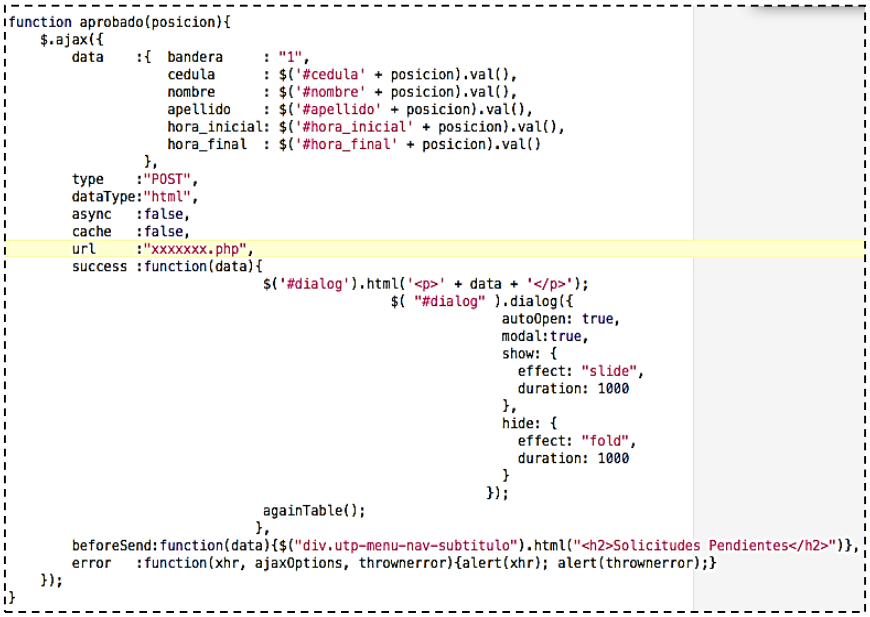
```
while ($row = @mysql_fetch_array($result)) {
    if($row["aprobada"] == '0'){
        echo '<tr class="style6">
                <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["usuario_cedula"] .'</span>
                    <input type="hidden" value="' . $row["usuario_cedula"] . '" id="cedula' . $i . '" name="cedu
                </td>
                <td  bgcolor="#FFFFFF"><span class="show">' . $row["Nombre_usuario"] . '</span>
                    <input type="hidden" value="' . $row["Nombre_usuario"] . '" id="nombre' . $i . '" name="nomb
                </td>
                <td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] . '</span>
                    <input type="hidden" value="' . $row["Apellido_usuario"] . '" id="apellido' . $i . '" name="
                </td>
                <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_inicial"] . '</span>
                    <input type="hidden" value="' . $row["hora_inicial"] . '" id="hora_inicial' . $i . '" name="
                </td>
                <td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_final"] . '</span>
                    <input type="hidden" value="' . $row["hora_final"] . '" id="hora_final' . $i . '" name="hora
                </td>
                <td  bgcolor="#FFFFFF"><span class="show">Pendiente</span>
                    <input type="hidden" value="' . $i . '" name="indentificador">
                </td>
                <td  bgcolor="#FFFFFF">
                    <input type="submit" value="Aprobar" id="' . $i . '" style=" width: 100%" name="aprobado' .
                    <br>
                    <input type="submit" id="' . $i . '" name="noaprobado' . $i. '" value="Rechazar" style=" wid
                </td>
            </tr>';
    }

    if($row["aprobada"] == '0'){


       $i++;
    }
}
```

Figura 112. Tabla con reporte de las solicitudes de acceso pendientes de aprobación.

Después de realizar la consulta, se recorre la información traída para incluirla en el cuerpo de la tabla con un encabezado definido. Por cada fila del reporte, en el código HTML de la Figura 112 se puede observar que la última columna contiene dos botones de aprobación y desaprobación de las solicitudes de un usuario para el control del Observatorio Astronómico de la Universidad Tecnológica de Pereira.



```
function aprobado(posicion){
    $.ajax({
        data    :{  bandera     : "1",
                    cedula      : $('#cedula' + posicion).val(),
                    nombre      : $('#nombre' + posicion).val(),
                    apellido    : $('#apellido' + posicion).val(),
                    hora_inicial: $('#hora_inicial' + posicion).val(),
                    hora_final  : $('#hora_final' + posicion).val()
                 },
        type    :"POST",
        dataType:"html",
        async   :false,
        cache   :false,
        url     :"xxxxxxx.php",
        success :function(data){
                                $('#dialog').html('<p>' + data + '</p>');
                                        $( "#dialog" ).dialog({
                                                autoOpen: true,
                                                modal:true,
                                                show: {
                                                  effect: "slide",
                                                  duration: 1000
                                                },
                                                hide: {
                                                  effect: "fold",
                                                  duration: 1000
                                                }
                                              });
                                againTable();
                                },
        beforeSend:function(data){$("div.utp-menu-nav-subtitulo").html("<h2>Solicitudes Pendientes</h2>")},
        error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
    });
}
```

Figura 113. Código Ajax para proceso de aprobación o desaprobación de acceso al control del observatorio.

Al dar clic en el botón de aprobación o desaprobación de la Figura 112 , se realiza una petición Ajax de Jquery como se puede notar en la Figura 113, la cual envía los datos de cada columna como; “Cédula”, “Nombre”, ”Apellidos”, ”Hora Inicial”, ”Hora final” de tal manera que puedan ser tratados posteriormente.

Previamente se consulta la tabla de tiempos solicitados en la base de datos, de modo que se reconozcan los posibles cruces de horarios solicitados entre los diferentes usuarios. Al dar clic sobre el botón “Aprobar” de la Figura 112, se despliega un cuadro de dialogo como el de la Figura 114 con una tabla de los cruces existentes con el registro que se desea aprobar, de modo que el administrador finalmente decide a quien le asigna cada una de las solicitudes enviadas.

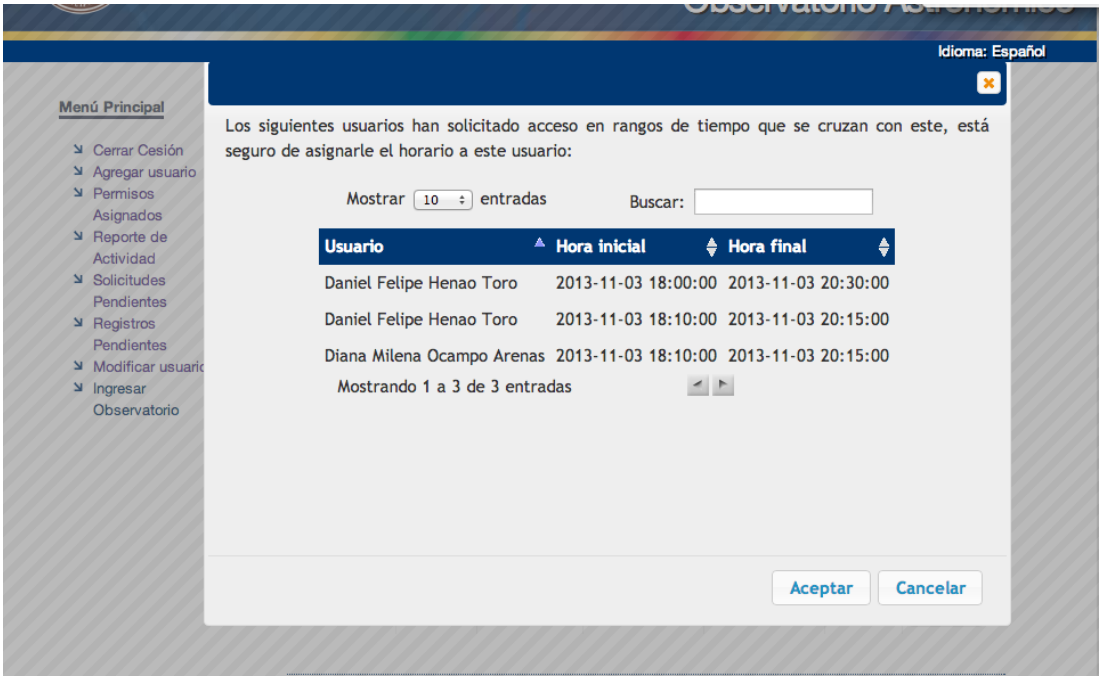


Figura 114. Cuadro de dialogo donde informa los cruces de horario en las solicitudes realizadas por los usuarios.

```
$sql    = "INSERT INTO `tiempo_permitido`(`usuario_Cedula`, `hora_inicial`, `hora_final`, `logueado`)
                VALUES ('" . $cedula . "','" .$hora_inicial . "','" . $hora_final ."','" . $logueado . "')";
mysql_query($sql) or die("Error en la creacion. MySQL dice:".mysql_error());
```

Figura 115. Sentencia SQL para la inserción de datos en la tabla “tiempo_permitido” cuando existe aprobación del administrador.

```
$sql    = "UPDATE `tiempo_solicitado` SET `aprobada`='" . $logueadoDos . "'
                WHERE `usuario_cedula`='" . $cedula ."' AND `hora_inicial`='" . $hora_inicial . "'
                    AND `hora_final`='" . $hora_final . "'";
```

Figura 116. Sentencia SQL que actualiza valores en la tabla “tiempo permitido”.

Cuando los datos de solicitud son aprobados se realiza una inserción en la tabla “tiempo permitido” como se muestra en Figura 115 y en la Figura 116, se actualiza sobre la tabla “tiempo solicitado” el rango de horas solicitado por el usuario.

```
$sSQL   = "SELECT Email_usuario FROM usuario WHERE Cedula_usuario='$cedula'";
$result = mysql_query($sSQL);
$row    = mysql_fetch_array($result);
$email  = $row["Email_usuario"];
mail($email,"Autorizacion ingreso a Observatorio UTP","Señor(a) " . $nombre . " " . $apellido . ", Se le ha autorizado .....
```

Figura 117. Código en PHP para el envío de correo eléctrico al usuario, informando sobre la aprobación de control vía web del observatorio.

Al ser exitosa la asignación del horario solicitado por el usuario, se envía un correo electrónico que indica la autorización del ingreso al control vía web del Observatorio Astronómico de la Universidad Tecnológica de Pereira como se puede observar en la Figura 117.

```
/* -- Ejecucion de la sentencia SQL INSERT -- */
$sql="DELETE FROM `tiempo_solicitado` WHERE `usuario_cedula`='$cedula'
        AND `hora_inicial`='$hora_inicial' AND `hora_final`='$hora_final'";
mysql_query($sql) or die("Error en la creacion. MySQL dice:".mysql_error());
```

Figura 118. Sentencia SQL para la eliminación de datos de la base de datos cuando no es aprobado el acceso a la plataforma.

Cuando los datos de solicitud no son aprobados se realiza un borrado del registro seleccionado en la tabla “tiempo solicitado” según la Figura 118, de modo que solo quede el historial de la actividad real realizada sobre el observatorio.

```
$sSQL   = "SELECT Email_usuario FROM usuario WHERE Cedula_usuario='$cedula'";
$result = mysql_query($sSQL);
$row    = mysql_fetch_array($result);
$email  = $row["Email_usuario"];
mail($email,"Autorizacion ingreso a Observatorio UTP","Señor(a) " . $nombre . " " . $apellido . ", No ha sido autorizado .....
```

Figura 119. . Código en PHP para el envío de correo eléctrico al usuario informado sobre la no aprobación de control vía web del observatorio.

De forma paralela, en el código de la Figura 119 se envía un correo que indica la no autorización del ingreso al control del Observatorio Astronómico de la Universidad Tecnológica de Pereira.

Registros de Usuarios Pendientes por Aprobación:

Figura 120. Reporte de registro de usuarios pendientes por aprobación por parte del administrador.

De igual forma, en la pantalla de la Figura 120 se muestra una tabla con una particularidad adicional, consiste en aprobar o desaprobar la solicitud de los registros de usuarios que no se encuentran en registrados en el sistema (capítulo 5.2.4.2), igualmente que el ítem anterior la tabla tiene encabezados titulados como; “Cédula”, “Nombre”, ”Apellidos”, ”Institución”, ”Comentarios”, “Estado” y “Registrar”, los cuales detallan la información personal de los usuarios no registrados en la plataforma web, en el reporte de registros de usuarios pendientes a aprobarse, el código construye toda la presentación inicial, más la acción relacionada con la aprobación o desaprobación de ellas, ver código completo en Anexo 15.

```
$user   = $_SESSION['Id'] ;
$result = mysql_query("SELECT * FROM `solicitud_registro`");
```

Figura 121. Sentencia SQL que consulta la información de los usuarios que han solicitado registro en la plataforma web.

En la Figura 121, se ejecuta una consulta sobre la tabla “solicitud registro” para poder trabajar con todos sus campos, y lograr la construcción del reporte correspondiente.

```
<tbody>
<?php
    /* -- Mostramos los registros -- */
    $i  = 0;
    while ($row = @mysql_fetch_array($result)) {
        if($row["registrado"] == '0'){
            echo '<tr class="style6">
                    <td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["id"] .'</span>
                        <input type="hidden" value="' . $row["id"] . '" name="cedula" id="cedula' . $i . '">
                    </td>
                    <td  bgcolor="#FFFFFF"><span class="show">' . $row["nombre"] . '</span>
                        <input type="hidden" value="' . $row["nombre"] . '" name="nombre" id="nombre' . $i . '">
                    </td>
                    <td  bgcolor="#FFFFFF"><span class="show">' . $row["apellido"] . '</span>
                        <input type="hidden" value="' . $row["apellido"] . '" name="apellido" id="apellido' . $i . '">
                    </td>
                    <td  bgcolor="#FFFFFF"><span class="show">' . $row["institucion"] . '</span>
                        <input type="hidden" value="' . $row["institucion"] . '" name="institucion" id="institucion' . $i . '">
                    </td>
                    <td  bgcolor="#FFFFFF"><span class="show">' . $row["comentarios"] . '</span>
                        <input type="hidden" value="' . $row["comentarios"] . '" name="comentarios" id="comentarios' . $i . '">
                    </td>
                    <td  bgcolor="#FFFFFF"><span class="show">Pendiente</span>
                        <input type="hidden" value="' . $i . '" name="indentificador">
                        <input type="hidden" value="1" name="bandera"   id="bandera">
                    </td>
                    <td  bgcolor="#FFFFFF">
                        <input type="submit" value="Aprobar" id="' . $i . '" style=" width: 100%" name="aprobado' . $i. '" onClick="aproba
                        <br>
                        <input type="submit" id="' . $i . '" name="noaprobado' . $i. '" value="Rechazar" style=" width: 100%" onClick="de
                    </td>';
            $i++;
        }
    }
    @mysql_free_result($result);
?>
</tbody>
```

Figura 122. Código para la Construcción de la tabla del reporte de usuarios pendientes por aprobación de registro en la plataforma.

Luego de construida la tabla, se recorren los datos obtenidos en la consulta anterior para incluirlos en el cuerpo con un encabezado definido, en la última columna correspondiente al encabezado “aprobar” se agregan dos botones visualizados en la Figura 122, uno corresponde a la aprobación y el otro a la desaprobación del registro solicitado por el usuario.

```
function aprobado(posicion){
    $.ajax({
        data    :{  bandera     : "1",
                    cedula      : $('#cedula' + posicion).val(),
                    nombre      : $('#nombre' + posicion).val(),
                    apellido    : $('#apellido' + posicion).val(),
                    pass        : $('#pass' + posicion).val()
                 },
        type    :"POST",
        dataType:"html",
        async   :false,
        cache   :false,
        url     :"xxxxxxxx.php",
        success :function(data){
                                $('#dialog').html('<p>' + data + '</p>');
                                            $( "#dialog" ).dialog({
                                                        autoOpen: true,
                                                        modal:true,
                                                        show: {
                                                          effect: "slide",
                                                          duration: 1000
                                                        },
                                                        hide: {
                                                          effect: "fold",
                                                          duration: 1000
                                                        }
                                                      });
                                againTable();
                               },
        beforeSend:function(data){$("div.utp-menu-nav-subtitulo").html("<h2>Registros Pendientes</h2>")},
        error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
    });
}
```

Figura 123. Código en Ajax para el proceso de aprobación o desaprobación del registro de usuario.

Al pulsar el botón de aprobación o desaprobación, en la Figura 123 se realiza una petición con Ajax de Jquery, enviando los datos de cada columna, como lo son: “Cédula”, “Nombre” y “Apellidos”, de tal manera que puedan ser tratados posteriormente.

```
$sql        = "INSERT INTO `usuario`
                (`Cedula_usuario`, `Nombre_usuario`, `Apellido_usuario`, `Email_usuario`, `Password_usuario`, `Institucion`)
                    SELECT `id`, `nombre`, `apellido`, `email`, `Password_usuario`, `institucion`
                        FROM `solicitud_registro` WHERE id = '" . $cedula . "'";

mysql_query($sql ) or die("Error en la creacion. MySQL dice:".mysql_error());

$sql        = "UPDATE `solicitud_registro` SET `registrado`='" . $logueadoDos . "' WHERE `id`='" . $cedula . "'";

mysql_query($sql ) or die("Error en la creacion. MySQL dice:".mysql_error());
```

Figura 124. Sentencia SQL para la inserción de datos de usuario al que se le aprueba el registro.

Cuando los datos de registro son aprobados se realiza una inserción de todos los campos en la tabla “usuario” y se actualiza el estado de la solicitud sobre la tabla “solicitud registro” como se puede apreciar en la Figura 124, de modo que se habilita el acceso a la plataforma al usuario que solicitó el registro.

```
$sSQL       = "SELECT Email_usuario FROM usuario WHERE Cedula_usuario='" . $cedula . "'";
$result     = mysql_query($sSQL);
$row        = mysql_fetch_array($result);
$email      = $row["Email_usuario"];

mail($email,"Registro en el Observatorio de la UTP","Señor(a) " . $nombre . " " . $apellido . ", Se le ha autorizado .....
```

Figura 125. Código PHP para el envío de correo al usuario que solicitó registro en la plataforma.

Al ser registrado el usuario en la plataforma web, en el código de la Figura 125 se envía un correo que le indica al usuario la aprobación de su ingreso a la plataforma.

```
$sql        =       "DELETE FROM `solicitud_registro` WHERE `id`='" . $cedula . "'";
mysql_query($sql) or die("Error en la creacion. MySQL dice:" . mysql_error());
echo "Usuario no aceptado";



mail($email,"Registro en el Observatorio de la UTP","Señor(a) " . $nombre . " " . $apellido . ", No ha sido autorizado .....
```

Figura 126. Sentencia SQL para la eliminación de datos innecesarios cuando se desaprueba la autorización de ingreso.

Cuando la solicitud de registro no es aprobada, en la Figura 126 se realiza una eliminación del registro correspondiente en la tabla “solicitud registro”. Además se envía un correo electrónico al usuario indicándole la no autorización del ingreso a la plataforma.

5.2.7. Página de Control Vía Web del Observatorio

Para el acceso a la página de control vía web del observatorio, debe existir una previa solicitud por parte del usuario (capítulo 5.2.5.3.) además de la respectiva aprobación por parte del administrador de la plataforma (capítulo 5.2.6.1.), para realizar la validación de este proceso se realiza el siguiente procedimiento:

El código completo que permite este proceso puede ser visualizado en el Anexo 16.

```
$user                  = $_SESSION['Id'] ;
$row['Nombre_usuario']  = $_SESSION['nom'] ;

$result = mysql_query("SELECT COUNT( * ) c FROM tiempo_permitido WHERE usuario_Cedula =$user AND hora_inicial <= NOW( )
                       AND hora_final >= NOW( )") or die("Couldn't query the user-database.");

$num = mysql_fetch_array($result);
$_SESSION["autentificado"] = "SIA";
if (!$num["c"]){
    header("Location: index.php");
}
```

Figura 127. Sentencia SQL para consultar el horario disponible para el acceso al control vía web del Observatorio.

Según la Figura 127, se consulta sobre la tabla “tiempo permitido” la cantidad de datos existentes allí con la condición de que si el horario actual en el que ingresa un usuario o administrador coincide con alguno registrado en la base de datos, el usuario logrará ingresar a la página de control del Observatorio, de lo contrario será redirigido a la página de inicio de la plataforma.

Figura 128. Página de control del observatorio con funcionalidades separadas en pestañas.

Como se puede observar en la Figura 128 al ingresar a la plataforma el usuario puede tener acceso a las diferentes funcionalidades que controlan los instrumentos básicos del Observatorio Astronómico. Esto se logra por medio de diversas tecnologías que permanecen en constante conexión con el servidor.

A continuación se detallan cada una de esas herramientas de control remoto que se visualizan como pestañas independientes en la página web de control:

5.2.7.1. Video Interno del Observatorio en Streaming

Este tipo de tecnología funciona mediante un búfer de datos que va almacenando lo que se va descargando en el dispositivo del usuario y de forma paralela le permite la visualización del contenido transmitido. El código necesario para la construcción de este flujo de video, puede ser visualizado en el Anexo 17.

El servicio de video streaming [24] se agrega mediante el uso del codificador Media Live Encoder desarrollado por la compañía Adobe, quien toma los datos enviados por una cámara a través de una tarjeta importadora de video. Estos datos son enviados al servidor Streaming Media Server desarrollado por la misma compañía, este se encarga de transmitir el audio y el video por sockets hacia un objeto HTML embed. Ver diagrama de conexión en la Figura 129.

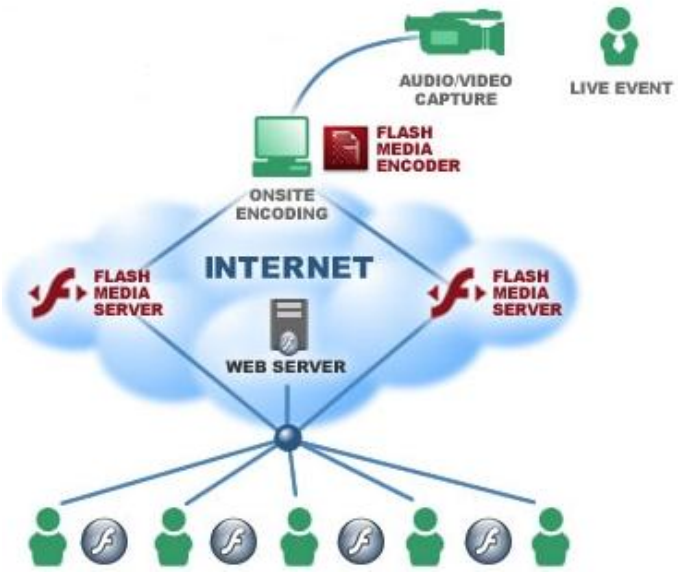


Figura 129. Diagrama de transmisión de video en Streaming.

Flash Media Live Encoder: El software de captura de audio y vídeo en directo Adobe Flash Media Live Encoder es un codificador multimedia que transmite audio y vídeo en tiempo real al software Adobe Media Server o a Flash Video Streaming

Service (FVSS). Este software puede habilitar la emisión de eventos en directo a través de redes locales o mediante la web.

Flash Media Server: Media Server permite a los desarrolladores distribuir vídeos con calidad de alta definición a la mayor cantidad posible de público en cualquier dispositivo conectado a Internet mediante un flujo de trabajo optimizado. Permite distribuir contenidos multimedia con una transmisión de flujo continuo consistente y protegida, compatible con una amplia gama de dispositivos: tablets, dispositivos móviles, televisores conectados y ordenadores de sobremesa.

Media Server tiene una arquitectura cliente-servidor. El código de cliente está escrito en ActionScript o Objective-C El código del servidor está escrito en ActionScript.

Media Server Standard ofrece cuatro servicios de streaming: vivo, VOD (video on demand), livepkgr (HTTP streaming) y multidifusión (RTMFP).

Instalación y Configuración de Servidor Streaming

Para la operación de este servicio se realizó la instalación y configuración de Media Live Encoder en su versión 3.2 y Media Server 4 como se puede notar en la Figura 130. A continuación se describe este proceso:

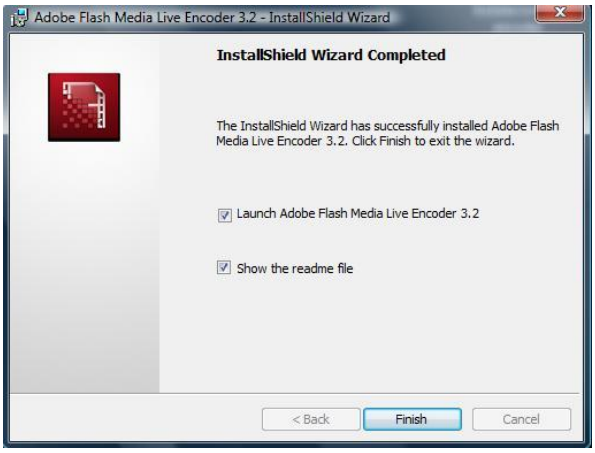


Figura 130. Instalación de la aplicación Live Encoder de Adobe.

La aplicación es instalada con todas sus propiedades como se encuentra por defecto y se configura la conexión al servidor Streaming Media Server 4, teniendo en cuenta la dirección a la que apunta y el nombre de la conexión.

FMS URL: rtmp://localhost/myLiveApp/instance1

Stream: mylivestream

Se eligen los dispositivos de audio y video conectados y se configura la calidad de la transmisión de estos datos, con las características que se pueden apreciar en la Figura 131.

Figura 131. Configuración de calidad de transmisión.

Posteriormente se inicia el proceso de captura de video y transmisión de datos hacia el servidor como se aprecia en la Figura 132.

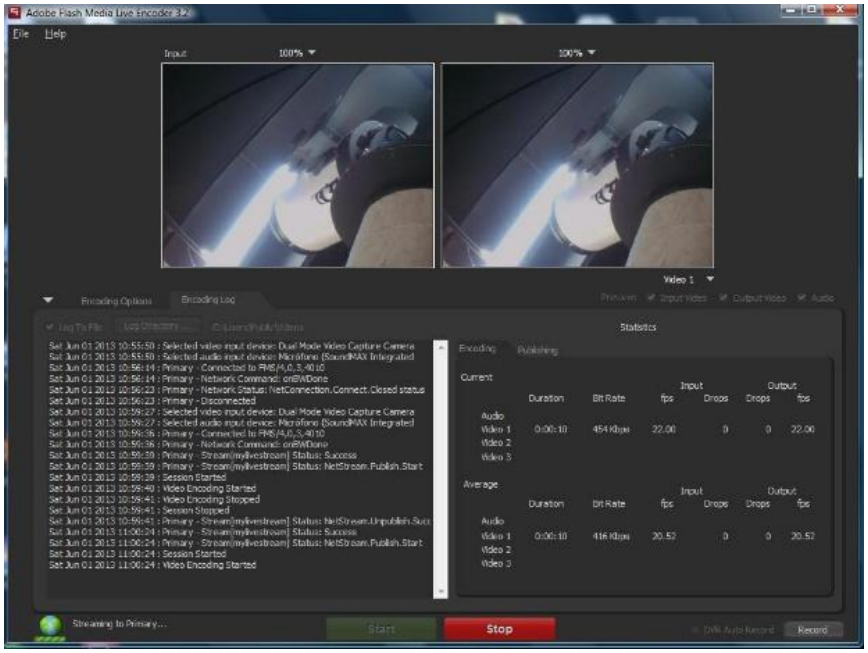


Figura 132. Aplicación Live Encoder de Adobe en ejecución.

Posteriormente, en la Figura 133 se muestra como se instala y configura la aplicación Streaming Media Server de Adobe teniendo en cuenta una adecuada elección de los puertos para la transmisión de datos.

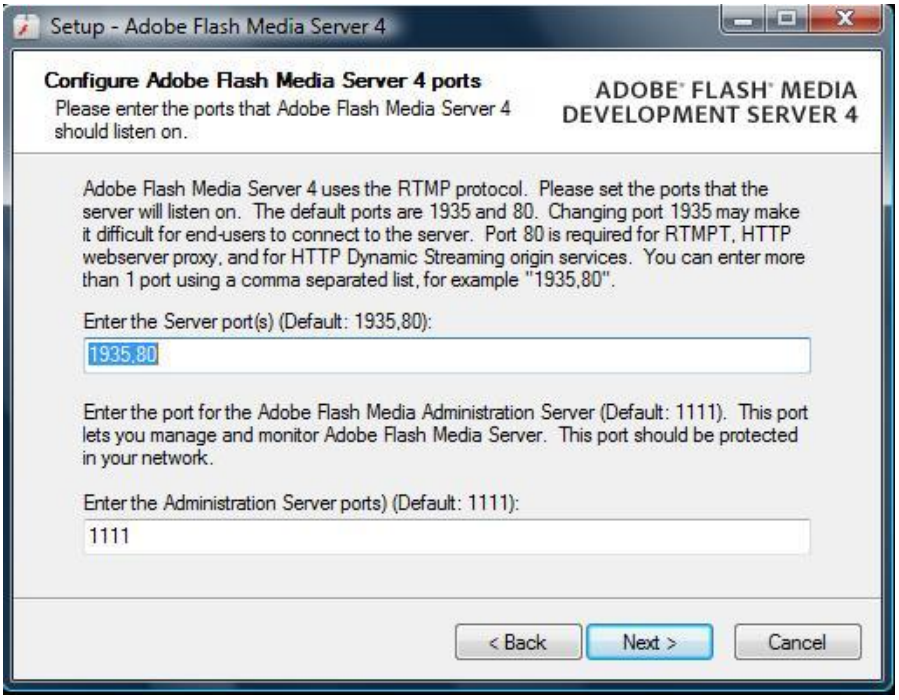


Figura 133. Configuración de puertos de Media Server de Adobe.

Luego se crea una nueva conexión llamada myLiveApp en la ruta C:\Program Files \Adobe\Flash Media Server 4\applications, desde la consola de administración de Media Server que se visualiza en la Figura 134.

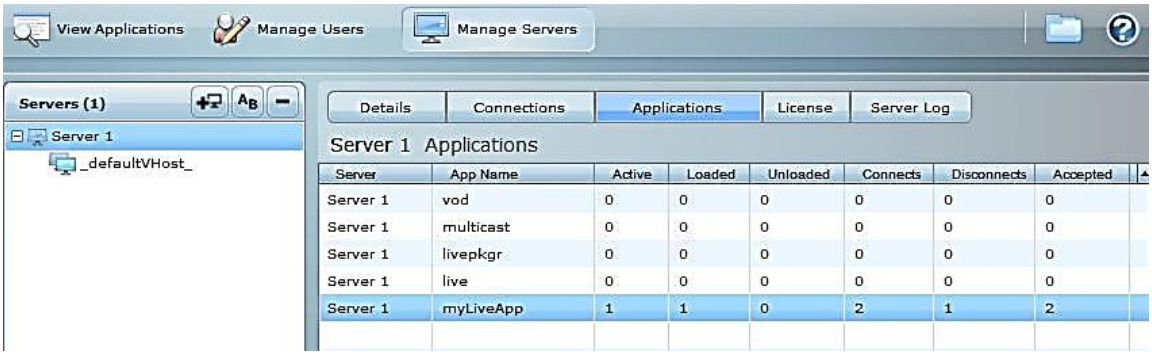


Figura 134. Consola de administración de Media Server de Adobe.

En esta consola también se puede apreciar el estado de la conexión, la cantidad de usuarios conectados y la cantidad de recursos utilizados por el sistema para la ejecución de las tareas como se aprecia en la Figura 135.

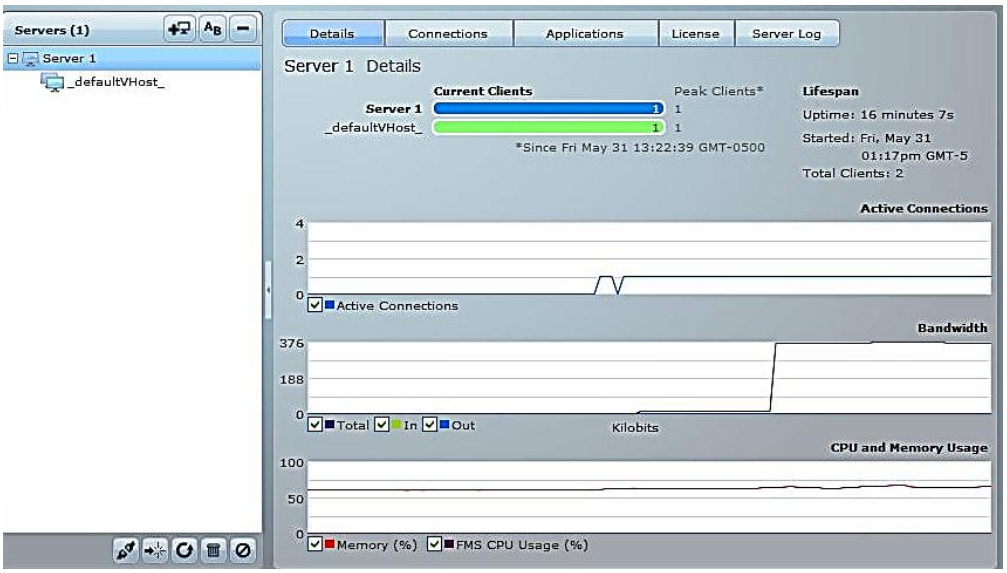


Figura 135. Detalles de estado de conexión de Media Server de Adobe.

Luego de realizada la instalación de las aplicaciones de Adobe, se procede a insertar la visualización del flujo de video en un elemento object de HTML 5 y se adecua a la tecnología multimedia sobre un navegador web como se puede apreciar en la Figura 136. El código completo para la inserción de este componente puede ser visualizado en el Anexo 18.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="342" height="291" align="middle" id="FLVPlayer">
  <param name="movie" value="FLVPlayer_Streaming.swf" />
  <param name="quality" value="high" />
  <param name="wmode" value="opaque" />
  <param name="scale" value="noscale" />
  <param name="salign" value="lt" />
  <param name="FlashVars" value=
"&amp;MM_ComponentVersion=1&amp;serverName=200.21.217.75&amp;skinName=Halo_Skin_3&amp;appName=myLiveApp/instance1&amp
mp;autoPlay=false&amp;autoRewind=true" />
```

Figura 136. Código para la inserción del flujo de video en página html.

5.2.7.2. Página de Control de Aplicaciones Mediante VNC

El protocolo VNC permite el control del escritorio de un servidor remoto, construyendo su imagen en el ordenador cliente por medio de puntos X y Y o pixel, y mediante la transmisión de los eventos del usuario que indican la manipulación del sistema servidor.

Para la realización de este componente se utilizó el sevidor ThinVNC SDK [25], el cual fue instalado y configurado como se describe a continuación:

Instalación y Configuración de Servidor VNC

Para el funcionamiento del servidor VNC como se ve en la Figura 137 solo se requiere de la instalación de ThinVNC SDK el cual se consigue descargándose de Cybele Software, el SDK se incluye en el paquete de instalación y contiene el desarrollo del servicio en diferentes lenguajes como Visual Basic .NET [26], Visual C++ y C# .

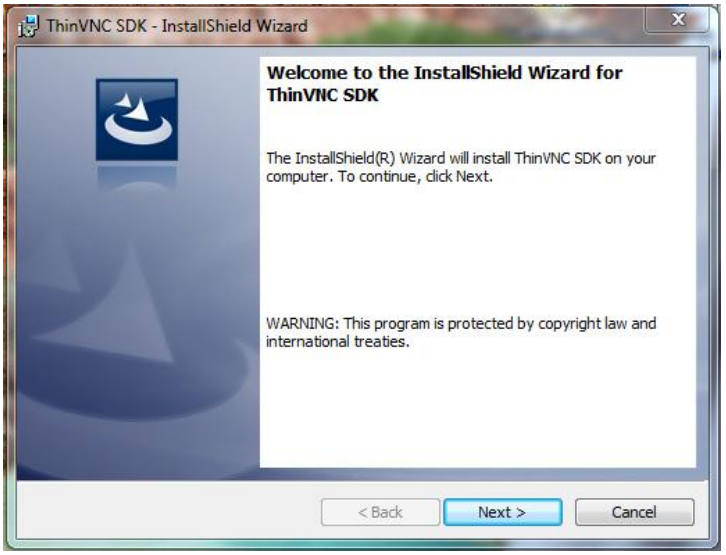


Figura 137. Inicio de instalación de ThinVNC SDK.

En la pestaña de control se muestra un botón de expansión a una ventana independiente donde se carga la imagen de tres aplicaciones que son usadas en el Observatorio Astronómico de la Universidad Tecnológica de Pereira, en la Figura 138 estas aplicaciones corresponden a Meade AutoStar Suite, EasyWeather Radio Controller y CCDOps5, el cual se encargan del movimiento del telescopio, la lectura climática y toma de fotografías respectivamente. El código completo que ejecuta el servicio VNC puede ser visto en el Anexo 19.

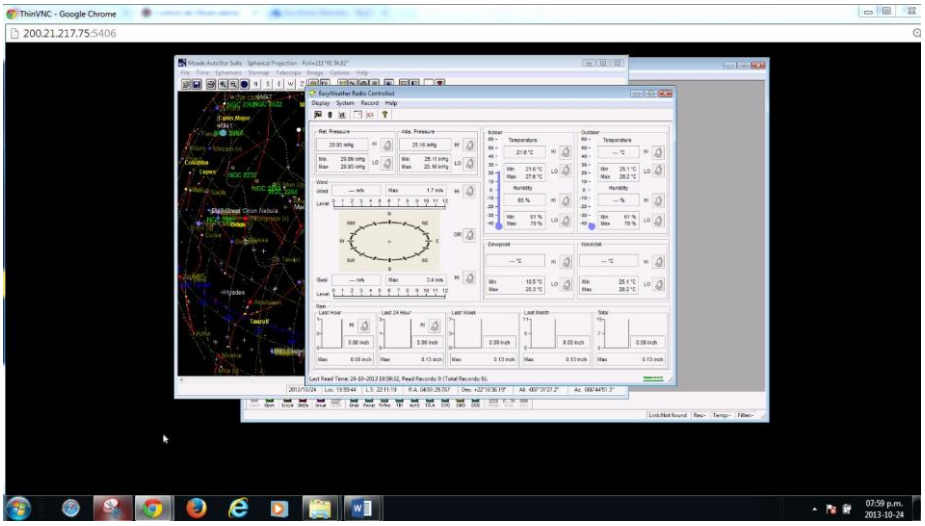


Figura 138. Página desplegada con aplicaciones corriendo sobre VNC.

En este código se logra configurar que aplicaciones y/o servicios pueden ser accedidos de forma remota, evitando que el usuario que accede al control mediante VNC, tenga permisos en la configuración del servidor, aplicaciones o control sobre el sistema operativo.

De acuerdo al código que se observa en la Figura 139, para que este servicio se ejecute de forma automática, se creó un archivo *.cmd, que se colgó al inicio del sistema del equipo de cómputo, utilizado como servidor.

```
@echo off
color 9F
mode con cols=60 lines=15
title "Inicializacion Automatica de Aplicaciones Control Via Web del Observatorio"

@echo off
echo Lanzando aplicacion 2
cd C:\Users\Public\Documents\ThinVNC SDK\samples\desktop\vb\ThinVncServer\bin\Release
echo aplicacion 2 lanzada
ThinVncServer.exe
```

Figura 139. Código para ejecución automática de módulo VNC.

5.2.7.3. Página de Control de Hardware del Observatorio.

En la pantalla de la Figura 140, la página de control de hardware contiene varios botones o controles sobre el hardware del Observatorio Astronómico, estos permiten al usuario enviar el valor lógico adecuado mediante el uso de la tecnología Node.JS – SerialPort, hacia el microcontrolador central que acciona los actuadores conectados a los dispositivos electrónicos, que finalmente permiten realizar las

siguientes acciones: abrir compuerta, cerrar compuerta, girar domo, detener domo", encender luz y apagar luz.

Figura 140. Página de control de hardware del Observatorio Astronómico.

Para el funcionamiento de esta tecnología, se realiza la instalación de Node.JS – SerialPort, la cual es una plataforma desarrollada sobre la máquina de ejecución de Javascript de Chrome para aplicaciones de fácil construcción y escalables en una red. Esta permite desarrollar aplicaciones orientadas a eventos sobre dispositivos distribuidos o del mismo sistema operativo donde se ejecuta. Por ello existen diversas variantes de enfoque de su uso, como:

Servidor web
Servidor de puerto serial
Servidor DNS
Gestión del sistema de archivos

La tecnología NodeJS permite desarrollar aplicaciones en lenguaje javascript ejecutadas sobre el servidor, en donde se pueden dar un plus adicional como usar herramientas del sistema operativo de la máquina, esto se logra por medio de la eficiencia en memoria local bajo altas cargas que se dividen por hilos de ejecución sobre el procesador y así atienden simultáneamente solicitudes multiusuario que se presentan en una aplicación de red.

Instalación y Configuración de Node.JS - SerialPort

El servicio de puerto serial provee su funcionalidad tras haber realizado la instalación de ciertos aplicativos que funcionan como intérpretes de lenguaje, en la Figura 141 se muestra el proceso de instalación del interprete Python 2.7.x, el cual posteriormente hace la función de compilación en el proceso de instalación del módulo serialport de Node.JS. Su configuración se muestra a continuación:

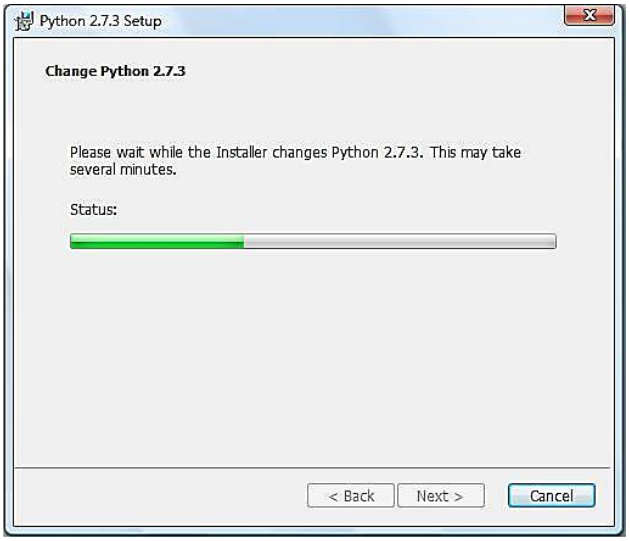


Figura 141. Instalación de interprete Python.

Luego de haberse instalado el intérprete python 2.7.x, se instala la plataforma Node.js basada en el procesador virtual de Javascript de Google, su instalación se realiza como se visualiza en la Figura 142.

Figura 142. Instalación de serialport.

Mientras se realiza el proceso de instalación de serialport, internamente python hace uso de librerías desarrolladas en C++, las cuales necesitan del framework .NET y su SDK instalados, además para obtener un proceso exitoso en tiempo de compilación se necesita el CLR (Common Language Runtime) [27] como ejecutor e intérprete de estas librerías, esto se consigue al instalar Visual Studio Express 2010, el proceso se muestra en la Figura 143.

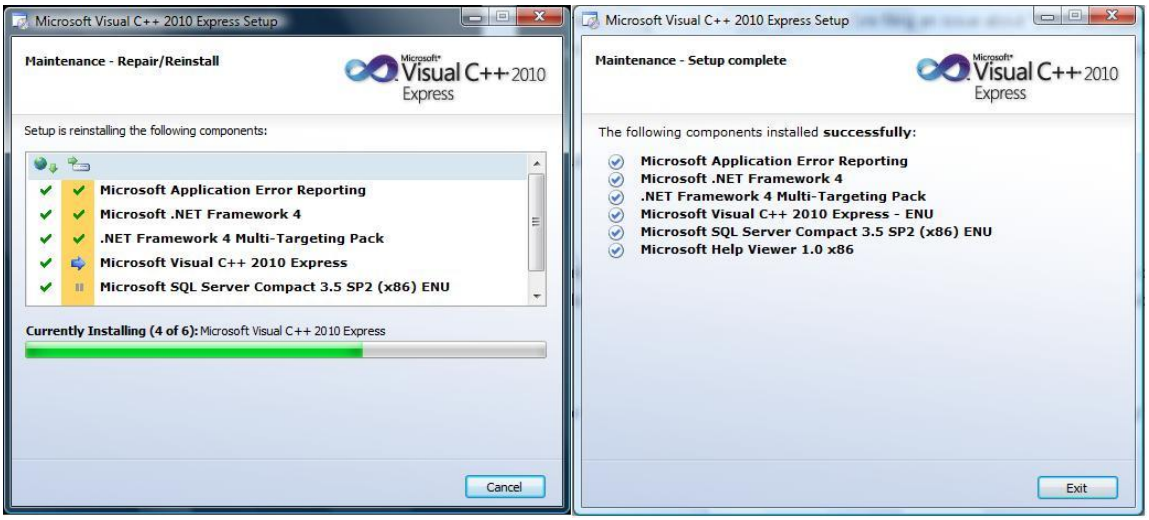


Figura 143. Instalación de Visual Studio Express 2010.

Finalmente, al tener instalados dichos aplicativos se procede a ubicarse en la dirección C:\xampp\htdocs\Observatorio\node_modules\ dentro de la consola de comandos y luego se ejecuta el comando npm install serialport [28], el proceso de instalación comienza a validar y compilar el módulo de serialport bajo los aplicativos anteriormente mencionados. El proceso se visualiza en la Figura 144.

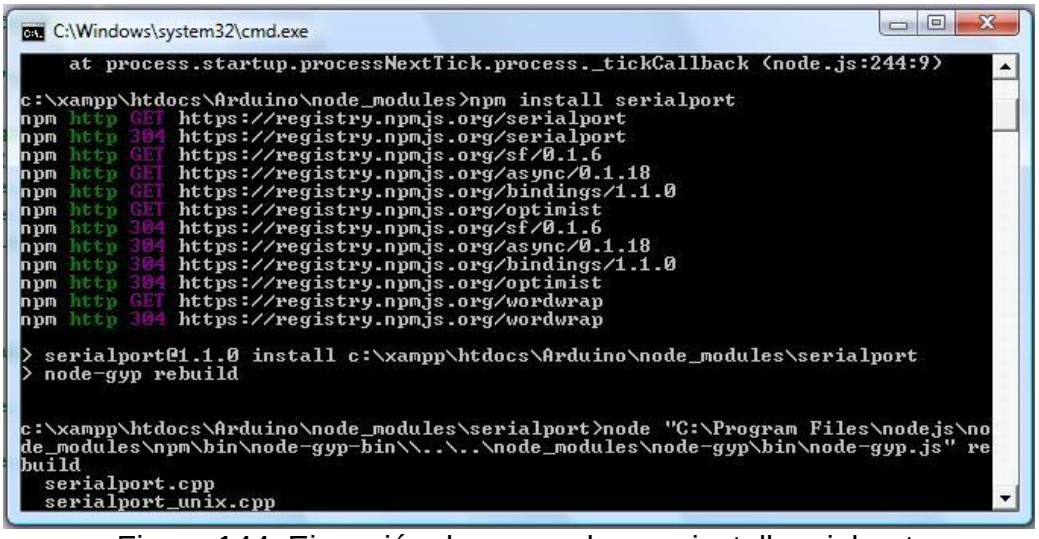


Figura 144. Ejecución de comando npm install serialport.

La página de control de dispositivos de hardware tiene una imagen constituida por iconos o botones creados a partir del código de la Figura 145.

```
<div id="fragment-4" style="width:700px;height:700px;">
    <div style="position:relative;left:130px;">
        <a id="line" class="selector"><img src="bulb.png" id="ligthON" class="ligth" onClick="enviar(this.id); bulb(this);" rel="ON"><
        <a id="line" class="selector"><img style="width:120px; height:127px;" id="domON" src="Earth_within_celestial_sphere.gif" onClic
        <a id="line" class="selector"><img style="width:120px; height:127px;" id="domOFF" src="ApagarDomo.png" onClick="enviar(this.id)
        <a id="line" class="selector"><img style="width:120px; height:127px;" id="openDOOR" src="AbrirCompuerta.png" onClick="enviar(th
    </div>
    <br>
    <div style="margin-left:150px; width:580px; height:200px;">
        <div style="width:590px; height:155px;" class="ui-state-hover" id="results">
            <div style="height:155px; overflow-y:scroll;" id="showResultAnt"></div>
        </div>
    </div>
</div>
```

Figura 145. Trozo de Código para la inserción de iconos en página de control de hardware.

```
function enviar(str){
        /*ESPERANDO RESPUESTA DESDE EL SERVIDOR Y CUANDO RECIBA EL EVENTO EMITIDO COMO "recibir",
        webSocket.on("recibir",recibirMensaje);
        webSocket.emit("enviar", str);
}
```

Figura 146. Función para envío de acciones desde los iconos de la página de control de hardware.

El hacer clic sobre cualquiera de estos iconos genera un evento de envío como se muestra en la Figura 146, el cuál notifica cuál de ellos se está ejecutando. Cada uno de estos eventos es interpretado en un sistema central, para ser enviados mediante puerto serial al microcontrolador central y conseguir el resultado final sobre el hardware.

```
var io          = require('socket.io').listen(xxxx);
var parser      = require("serialport").parsers;
var SerialPort  = require("serialport").SerialPort;
var serialPort  = new SerialPort('xxx');
var transmit    = false;
var datos       = false;
```

Figura 147. Configuración de puerto serial.

En el código de la Figura 147 alojado en el lado del servidor, se encuentra un script en ejecución donde se configura el puerto de escucha de la conexión con el cliente navegador y el puerto COM correspondiente al serial en comunicación con el hardware.

```
serialPort.on('open', function(){

            myCallBack('hardware',false,showCall);
            io.sockets.on("connection", arranque);

            serialPort.on('data', function(data) {

            ....
            /* IMPLEMENTACION DE SERIAL */

            }
}
```

Figura 148. Script de manipuladores de evento.

Como se muestra en el código de la Figura 148, el script posee dos manipuladores de evento, estos se encuentran relacionados con la apertura del puerto COM y la activación de datos presentes en el buffer. Cuando existe un dato en el buffer que proviene del puerto serial este se toma, se procesa y se reenvía hacia el cliente navegador.

```
var character = conmutarEstado('ENVIARUSUARIO', datos);
if(datos != false){
    console.log(character);
    serialPort.write(character);
}
```

Figura 149. Función "write".

En la Figura 149, se usa la función "write" del puerto serial para enviar datos hacia el mismo buffer, en el que quedan disponibles para la posterior recepción por el microcontrolador.

```
/*[SERIAL:: ANALISIS DATOS SERIAL]*/
recepcion   = parser.readline();
var str     = conmutarEstado('RECIBIRUSUARIO', recepcion('',data));
/*[/SERIAL:: ANALISIS DATOS SERIAL]*/
```

Figura 150. Función "readline".

En la Figura 150 se puede observar que se usa la función "readline" del puerto serial para recibir datos desde el mismo buffer y se puedan procesar y retransmitir hacia el cliente navegador.

5.2.8. Cerrar Sesión

El código de la Figura 151 describe la destrucción de la sesión que se encuentra creada por el usuario al momento de ingresar a la plataforma web, El código completo del cierre de sesión puede ser visualizado en el Anexo 20.

Este proceso realiza la actualización sobre la tabla “log” de la base de datos “bd” del campo “hora salida” por la fecha y hora actual, verificando que el usuario este activo y la hora de entrada coincida con la hora en la que el usuario inicialmente creo la sesión (capítulo 5.2.4.1).

```
$_SESSION       =   array();
session_destroy();

/* -- Se redirecciona a la página inicial index.php  -- */
echo '<script>
            window.location="index.php"
      </script>';
```

Figura 151. Código encargado de destruir las sesiones de usuario.

6

Pruebas y Resultados

Con el fin de verificar el adecuado funcionamiento de los elementos de hardware y software utilizados para el control vía web del Observatorio Astronómico de la Universidad Tecnológica de Pereira, se realiza cada uno de los pasos necesarios para que un usuario tenga acceso al control de la plataforma, incluyendo cada uno de los procesos y las respectivas validaciones existentes, ejecutadas por la plataforma, o realizadas por el administrador. De igual manera se verifica el funcionamiento de los dispositivos electrónicos que hacen posible dicho control.

6.1. Acceso a la Plataforma Web desde Equipo Remoto

Utilizando un equipo de cómputo con conexión a internet, se ingresa desde el navegador web Chrome a la siguiente URL: http://200.21.217.75/observatorio/, que corresponde a la IP Fija asignada y la ruta donde se aloja la información en el computador usado como servidor en el observatorio Astronómico de la UTP.

Figura 152. Página principal para control de acceso.

En la Figura 152 se visualiza como el navegador abre la página principal para el control de acceso a la plataforma, al hacer clic sobre la opción “registrarse” se obtiene un formulario que solicita los datos básicos del usuario como se describe en la Figura 153.

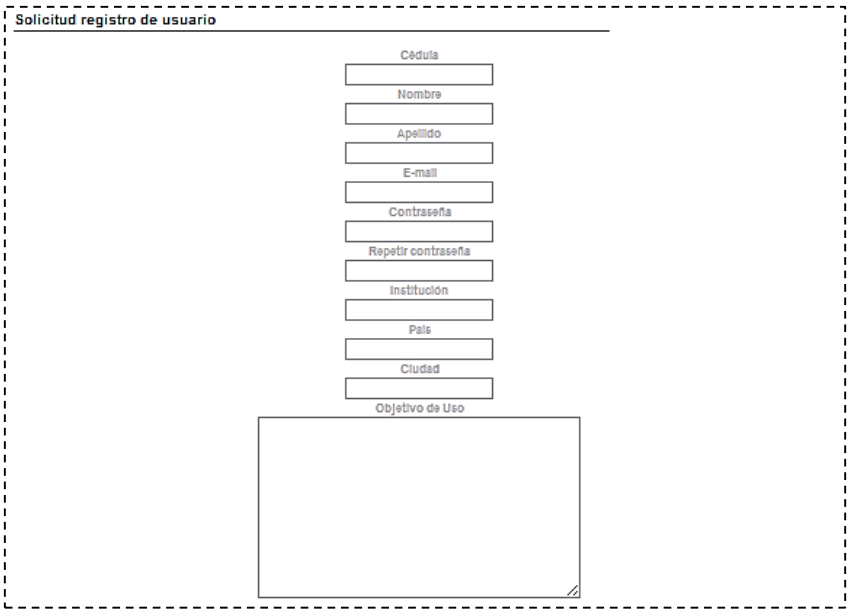


Figura 153. Formulario para solicitud de registro de un usuario.

Se realiza la operación de llenar el formulario, ingresando un número de identificación ya registrado en la plataforma, y se obtiene la validación de esta situación mediante el mensaje “Este número de identificación ya se encuentra Registrado como se observa en la Figura 154”.

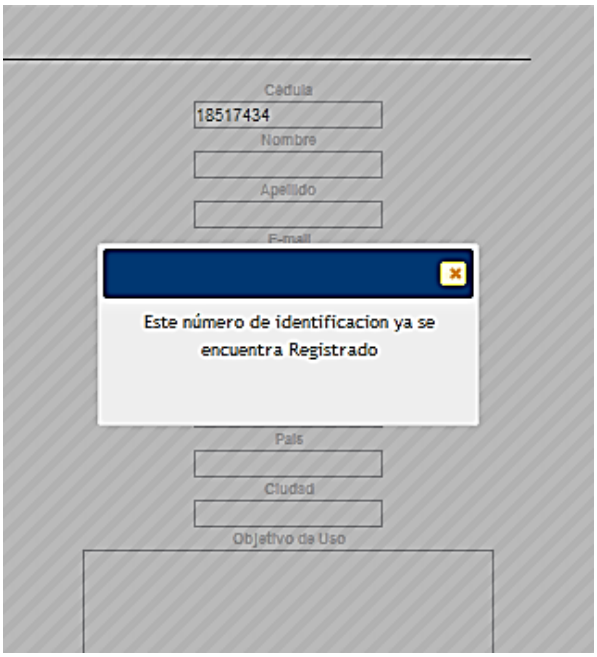


Figura 154. Validación de usuarios registrados.

Luego se envian los datos del formulario con al menos un campo vacío, y se logra la validación de esta situación mediante el mensaje “Datos incompletos, por favor inténtelo nuevamente” como se muestra en la Figura 155.

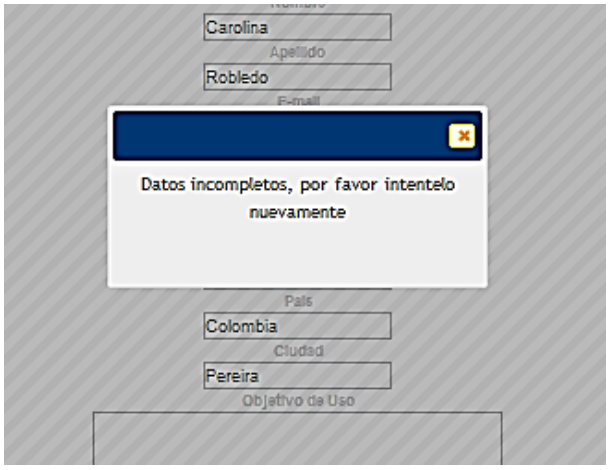


Figura 155. Validación de datos completos en el formulario.

Posteriormente se envian los datos del formulario con un formato de correo electrónico invalido, y se logra la validación de esta situación mediante el mensaje “Por favor ingrese un Email correcto” como se verifica en la Figura 156.

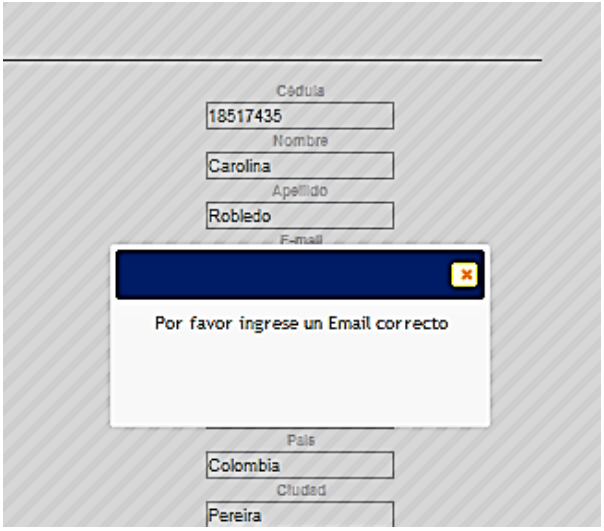


Figura 156. Validación de correo electrónico.

Luego, de verificar el funcionamiento adecuado de las validaciones del formulario, se incluye la información de forma correcta y se envían los datos registrados en él, recibiendo el mensaje visualizado en la Figura 157: “Solicitud enviada correctamente, esta información será validada por el administrador en un término menor a tres días hábiles, posteriormente recibirá un correo informando sobre el

estado de solicitud. Guarde su contraseña en un lugar seguro para poder ingresar luego de ser aprobado su registro.”.

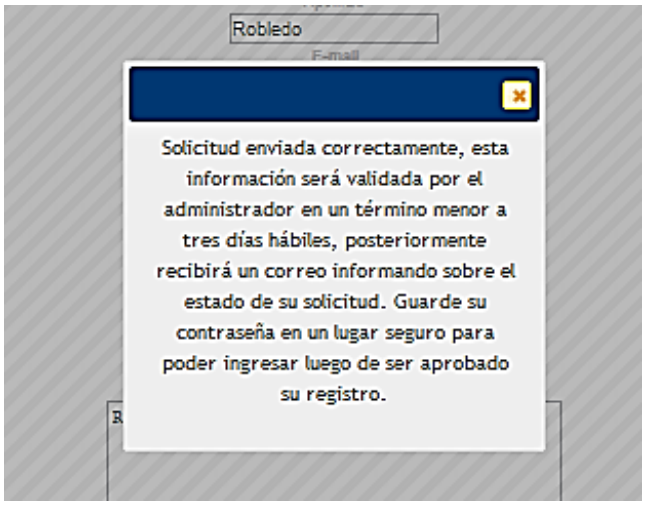


Figura 157. Mensaje para el usuario que solicitó registro dentro de la plataforma.

Nuevamente se realiza la solicitud con el mismo documento de identificación, y se, logra la validación de esta situación mediante el mensaje “Ya se encuentra una solicitud de registro pendiente” como se aprecia en la Figura 158.

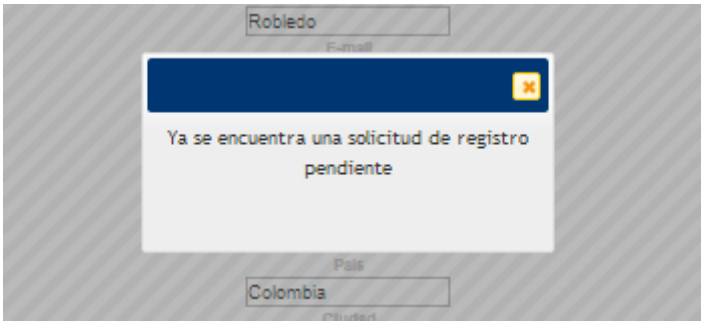


Figura 158. Validación de registro pendiente.

Como se observa en la Figura 159, luego de enviada la solicitud de registro, el administrador de la plataforma recibe mediante correo electrónico una notificación sobre este suceso, de este modo se entera que debe ingresar a la plataforma con el perfil de administrador para aprobar o desaprobar la solicitud de registro.

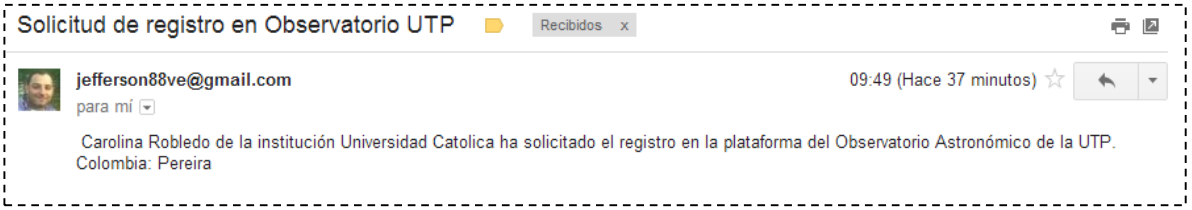


Figura 159. Mensaje recibido en la bandeja de entrada del correo electrónico del administrador.

Se ingresa a la plataforma con perfil de administrador y se verifica dentro de la opción de registros pendientes (Figura 160), el registro solicitado por parte del usuario, se evidencia que los datos incluidos en el formulario coinciden con los datos mostrados en el reporte. También se aprecian las opciones de aprobar o rechazar dicho registro.

Figura 160. Reporte de registros pendientes desde el perfil de administrador.

Se da aprobación al usuario y se obtiene la notificación “Usuario agregado correctamente”, como se visualiza en la Figura 161.

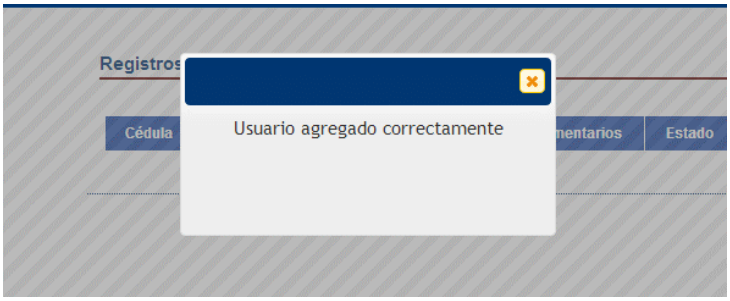


Figura 161. Mensaje del sistema informando que el usuario ha sido agregado.

Como se observa en la Figura 162, de forma paralela a este suceso, se verifica que el usuario ha recibido mediante correo electrónico, la notificación de que se le ha autorizado el registro como usuario de la plataforma web del observatorio astronómico y que puede ingresar con su número de identificación y la contraseña.

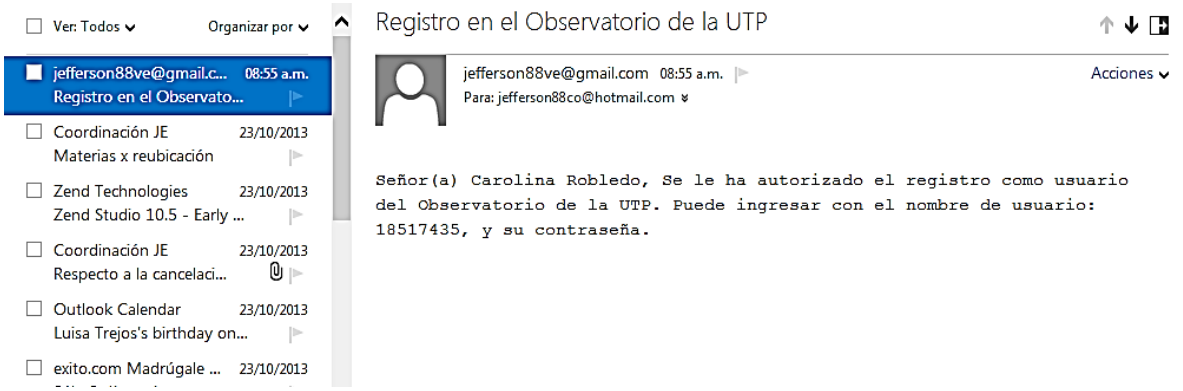


Figura 162. Mensaje recibido en la bandeja de entrada del correo electrónico del usuario.

Se ingresa con el perfil de usuario y se verifica el funcionamiento de cada una de las opciones presentes en el menú principal: Últimos accesos, editar datos, solicitudes realizadas, horarios no disponibles y solicitar acceso.

En la Figura 163 se visualiza la verificación de la opción “Últimos acceso” donde se evidencia que su ultimo acceso es del 24 de octubre de 2013 a las 10am, que coincide con la fecha y hora en la que realizó el ingreso, el registro de salida, aun no se evidencia, porque es la primera vez que ingresa y aún no ha cerrado la sesión.

Figura 163. Reporte de últimos acceso generado con el perfil de usuario.

Luego, se verifica la opción “Editar datos”, donde se abre el formulario para la edición de datos del usuario como el visualizado en la Figura 164, en este se adiciona el segundo apellido del usuario y se modifica la contraseña para el acceso a la plataforma.

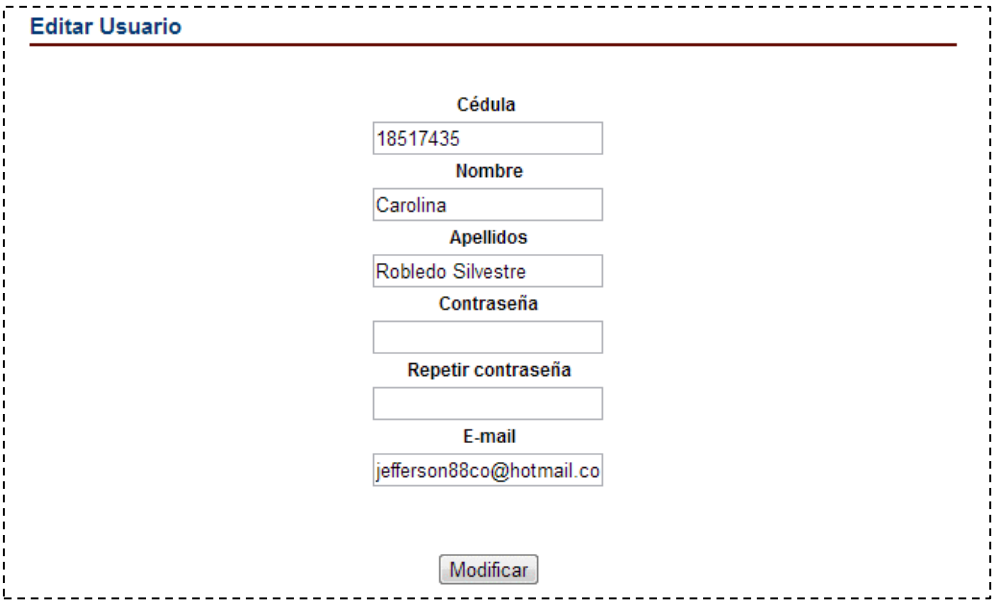


Figura 164. Formulario para editar datos del usuario.

Al hacer clic sobre el botón “Modificar” se obtiene el mensaje “Datos modificados correctamente” como el que se aprecia en la Figura 165.

El único dato que no puede ser modificado es el campo “Cédula”, este valor se muestra en el formulario, pero no permite ser modificado, teniendo en cuenta que este valor es una llave primaria de la base de datos “bd” y al ser modificado se pierde el historial de acceso a la plataforma web.

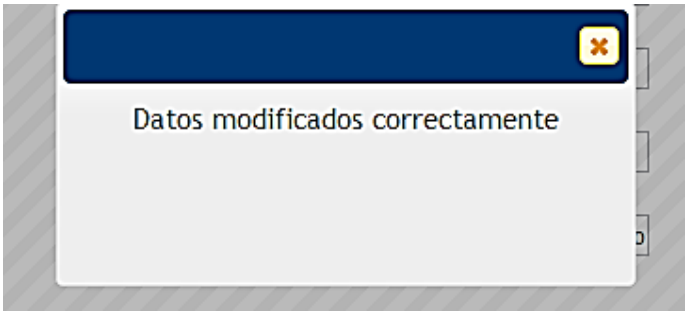


Figura 165. Mensaje del sistema, indicando que los datos han sido modificados.

Posteriormente, se realiza la verificación del funcionamiento del reporte de horarios no disponibles para el uso de control vía web del observatorio, en la Figura 166 se muestra el reporte donde se identifican las franjas de horario que otros usuarios ya han solicitado y han sido aprobadas por el administrador.

Figura 166. Reporte de horarios no disponible generado con el perfil de usuario.

Luego, se ingresa a la opción de solicitud de acceso a control desde el menú principal y se obtiene el formulario para realizar la solicitud como se muestra en la Figura 167.

Figura 167. Formulario para solicitud de acceso al control web del Observatorio.

Primero se hacen las validaciones lógicas para verificar el funcionamiento adecuado de la plataforma, lo primero es ingresar una fecha y hora de inicio superior a la final, donde se obtiene la validación del sistema "El horario final no puede ser menor o igual al inicial", como se muestra en la Figura 168.

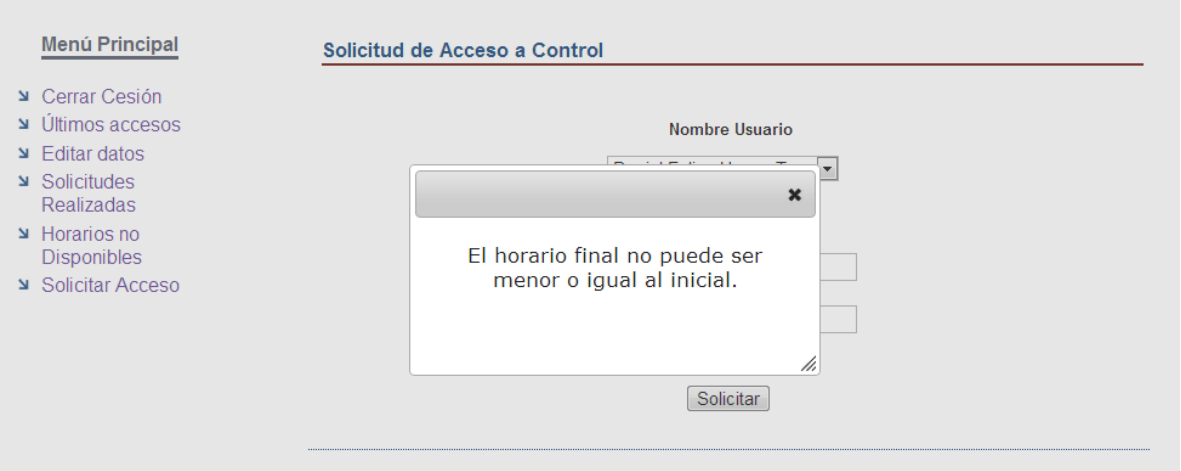


Figura 168. Mensaje del sistema, informando sobre validación de horario.

Finalmente, se envían los datos en un rango de tiempo adecuado, recibiendo la notificación "Solicitud enviada correctamente", como se visualiza en la Figura 169.

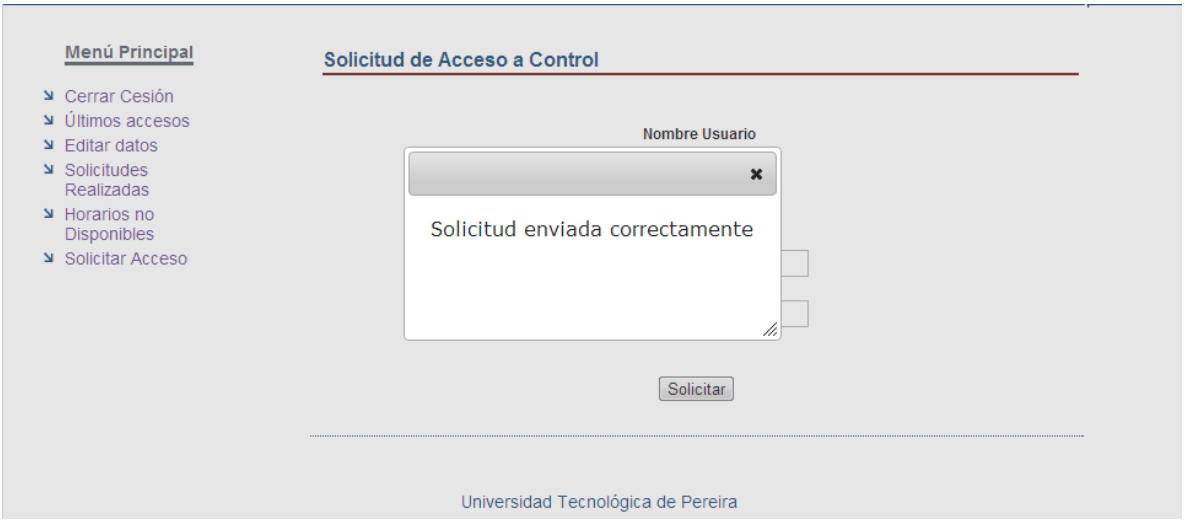


Figura 169. Mensaje del sistema, informando sobre la solicitud enviada.

Paralelamente a este evento se verifica que el administrador ha recibido una notificación mediante correo electrónico que le informa que el usuario ha realizado dicha solicitud como aparece en la Figura 170.

Figura 170. Bandeja de entrada de correo electrónico del administrador de la plataforma.

Posteriormente se verifica la opción “Solicitudes Realizadas”, donde también se observa que la solicitud realizada aún se encuentra en estado pendiente para su aprobación, como aparece en la Figura 171.

Figura 171. Reporte de solicitudes realizadas generado con el perfil de usuario.

Ahora se ingresa con el perfil de administrador con el fin de verificar el correcto funcionamiento de la aprobación de la solicitud realizada, para esto se elige del menú principal la opción "Solicitudes Pendientes", visualizando de esta manera el respectivo reporte, como se visualiza en la Figura 172.

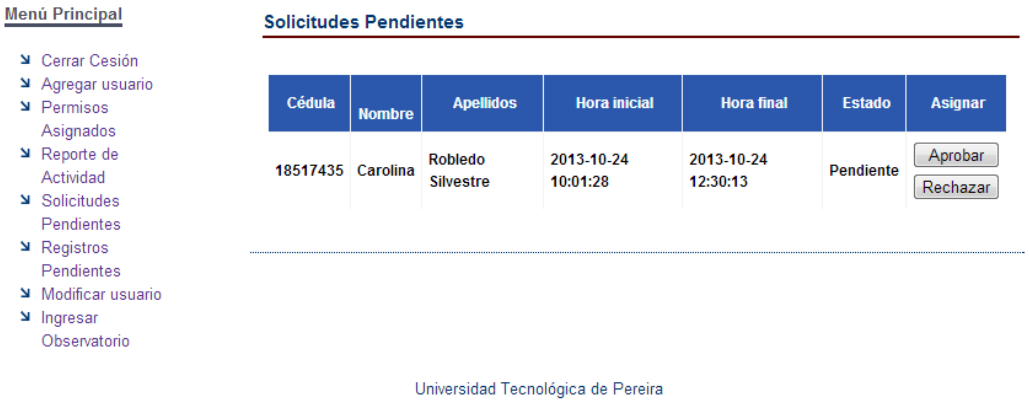


Figura 172. Reporte de solicitudes pendientes generado con el perfil de administrador.

En este reporte se encuentran las opciones de aprobar o desaprobar la solicitud de acceso realizada por el usuario, se marca la opción aprobar y se abre un cuadro de diálogo con los cruces de horario registrados o solicitados en el sistema como se

muestra en la Figura 173, de este modo el administrador puede verificar si este horario le puede ser asignado a este usuario o prefiere asignárselo a otro diferente.

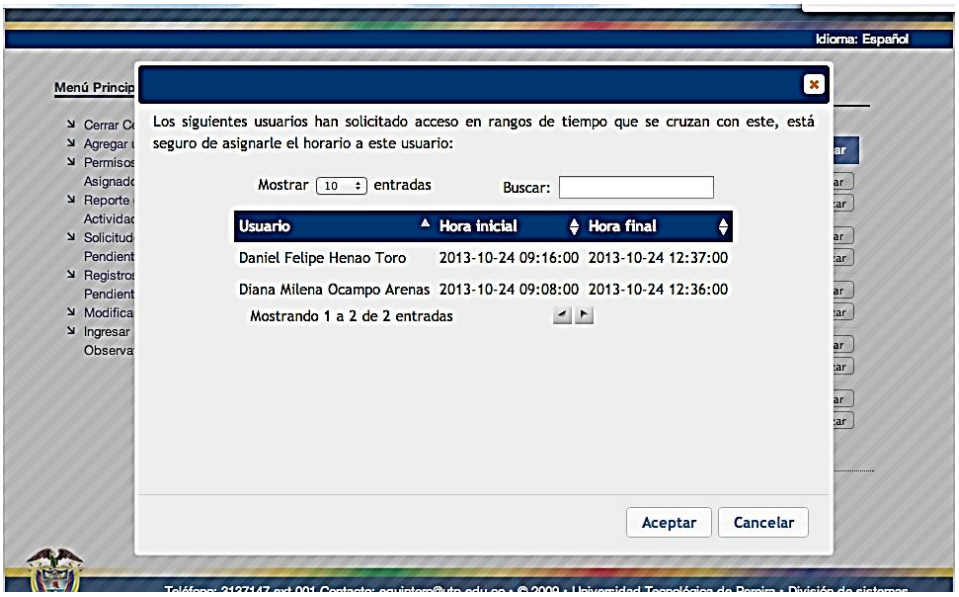


Figura 173. Cuadro de diálogo con datos de cruces de horario.

Luego, se selecciona el botón “Aceptar” de la Figura 173 y se recibe la notificación por parte del sistema con el mensaje "Horario asignado correctamente" como se observa en la Figura 174.

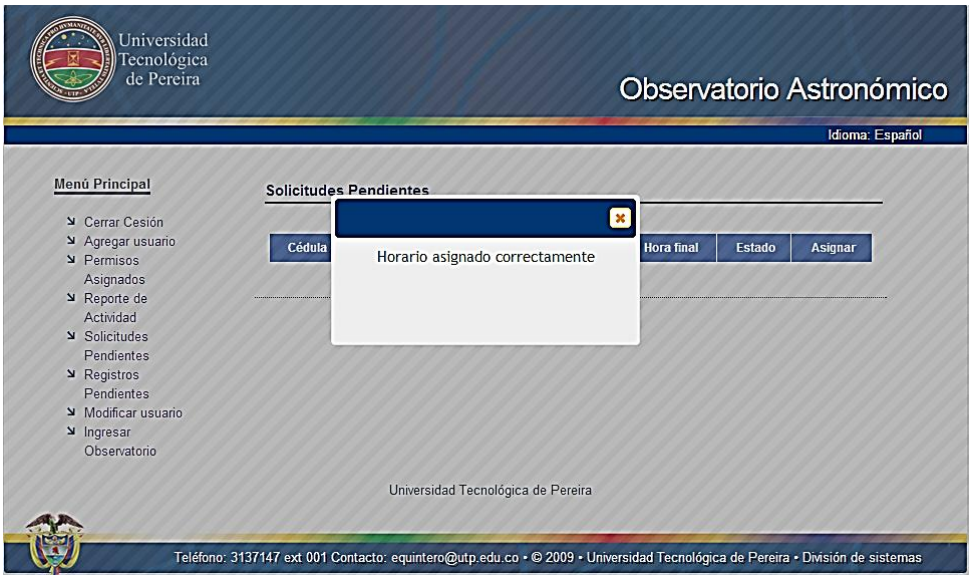


Figura 174. Mensaje del sistema, informando sobre asignación de horario para control vía web del observatorio.

En la Figura 175 se visualiza como paralelamente a este evento, se verifica que el usuario ha recibido esta notificación mediante correo electrónico, de modo que se logra enterar que ha sido aprobada su solicitud.

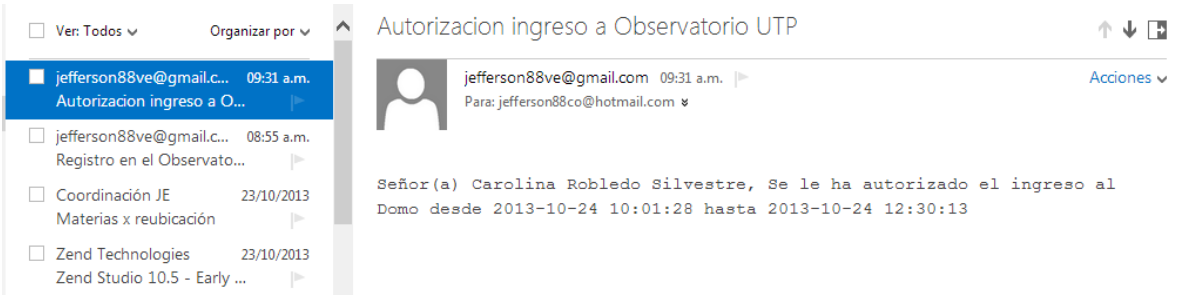


Figura 175. Bandeja de entrada de correo electrónico del usuario.

En la Figura 176 se muestra como nuevamente se verifica con el perfil de administrador, el reporte de permisos asignados, donde se evidencia que aparece asignada la franja horaria solicitada.

Figura 176. Reporte de permisos asignados generado con el perfil de administrador.

En la Figura 177 se visualiza como se verifica el reporte de los últimos acceso realizados a la plataforma web, eligiendo la opción “Reporte de Actividad” del menú principal, en este reporte el administrador puede verificar que usuarios y en que fechas y horarios han ingresado al sistema de control vía web.

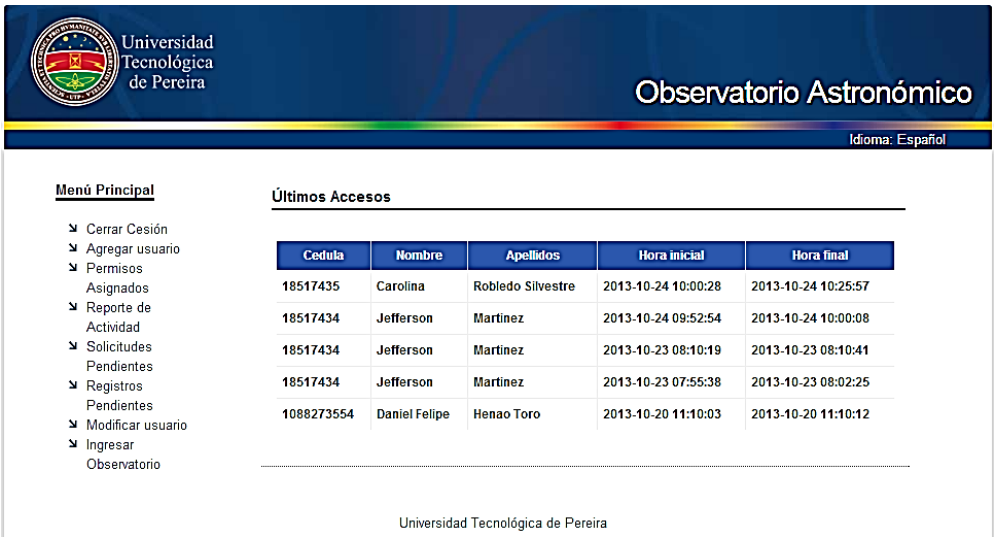


Figura 177. Reporte de últimos accesos generado con el perfil de administrador.

Después de aprobado el acceso al control vía web del observatorio, se accede nuevamente a la plataforma con el perfil de usuario, donde se puede visualizar que en el menú principal aparece la opción “Controlar Observatorio”, tal como se puede observar en la Figura 178, de este modo se verifica que la aceptación por parte del administrador se validó correctamente.

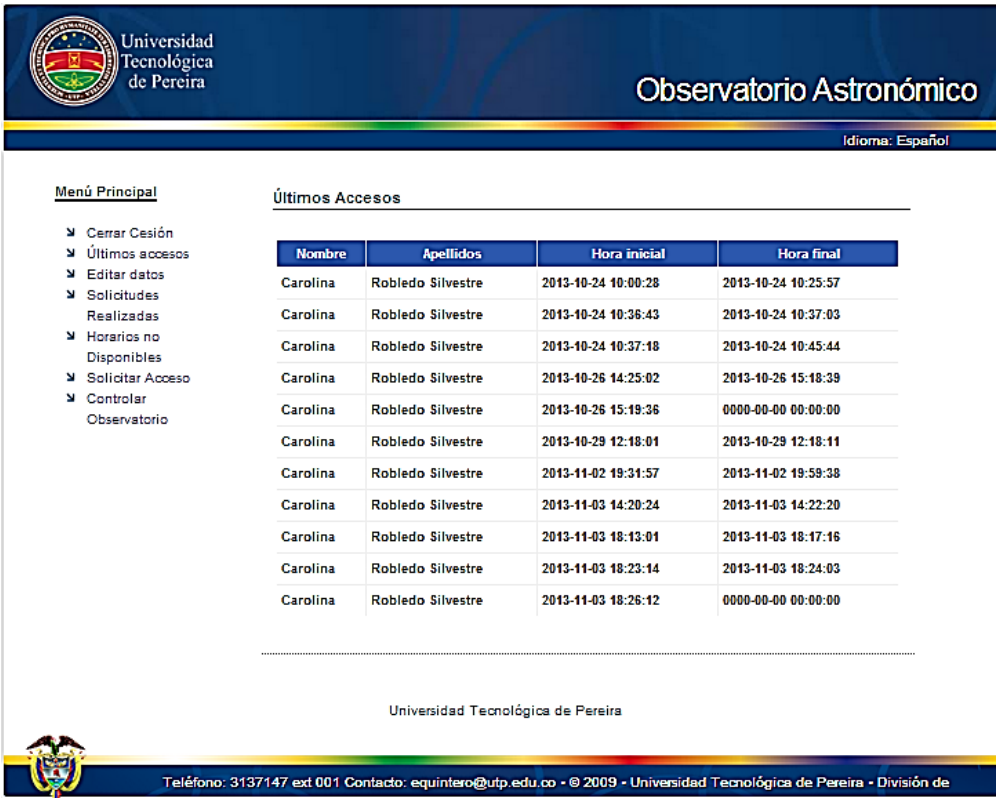


Figura 178. Reporte de últimos accesos generado con el perfil de usuario.

Luego, se ingresa a la opción "Controlar Observatorio" ya disponible en el menú principal, de esta forma se abre una nueva página con un menú en forma vertical (Figura 179) con las diferentes opciones para controlar el hardware disponible en el Observatorio.

En la Figura 180 se observa la primer pestaña de este menú "Video Interno", donde se muestra el video de lo que se observa en la dirección hacia la que apunte el telescopio, verificando de este modo el correcto funcionamiento de la transmisión de video en streaming dentro de la plataforma.

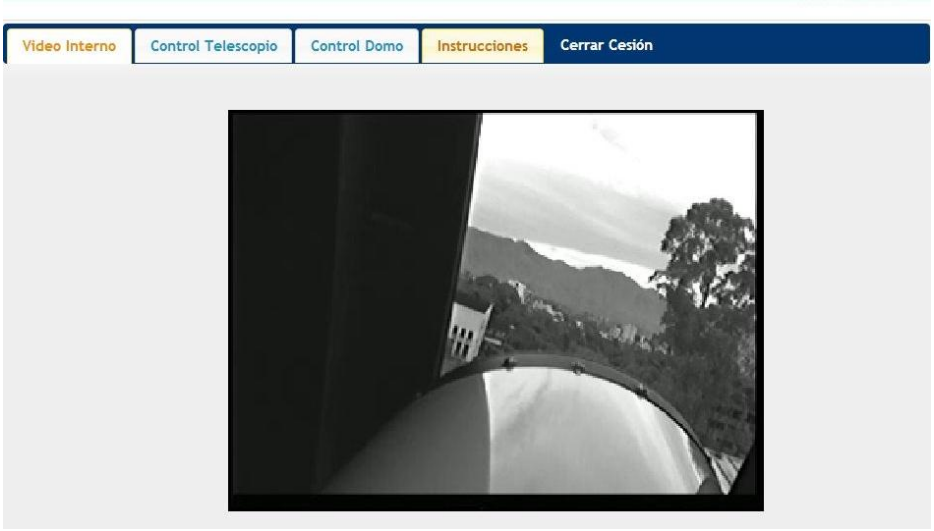


Figura 179. Video interno del Observatorio en señal streaming.

En la Figura 180 se visualiza la segunda pestaña "Control telescopio" donde se presenta un botón con la opción de expandir ventana, se hace clic sobre esta opción y se abre una nueva ventana con 3 aplicaciones para el control del observatorio (AutoStar, EasyWeather y CCDOps).

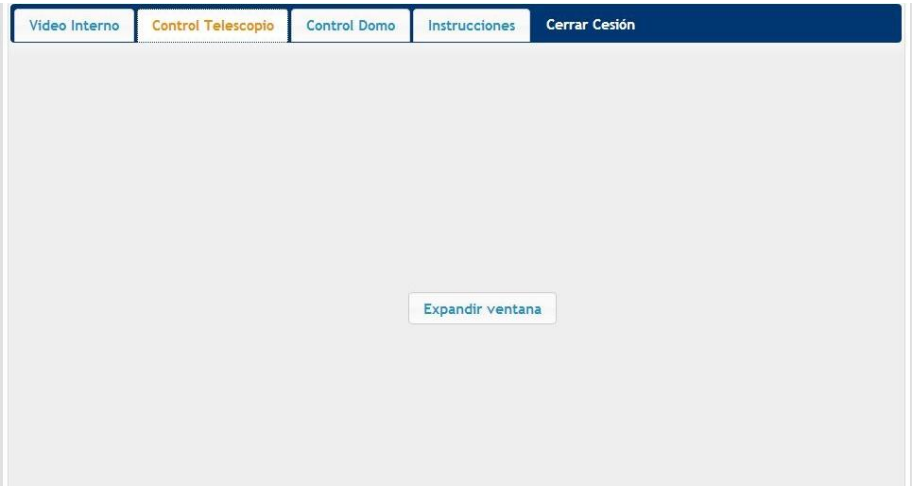


Figura 180. Pestaña de acceso a Control de aplicaciones mediante VNC.

En la Figura 181 se visualiza el adecuado funcionamiento de la conexión VNC desde el servidor web, además de que se encuentra en ejecución el servicio desarrollado en Visual Basic para ejecutar solamente los tres programas requeridos, limitando el acceso a otras aplicaciones o configuraciones del equipo de cómputo utilizado como servidor.

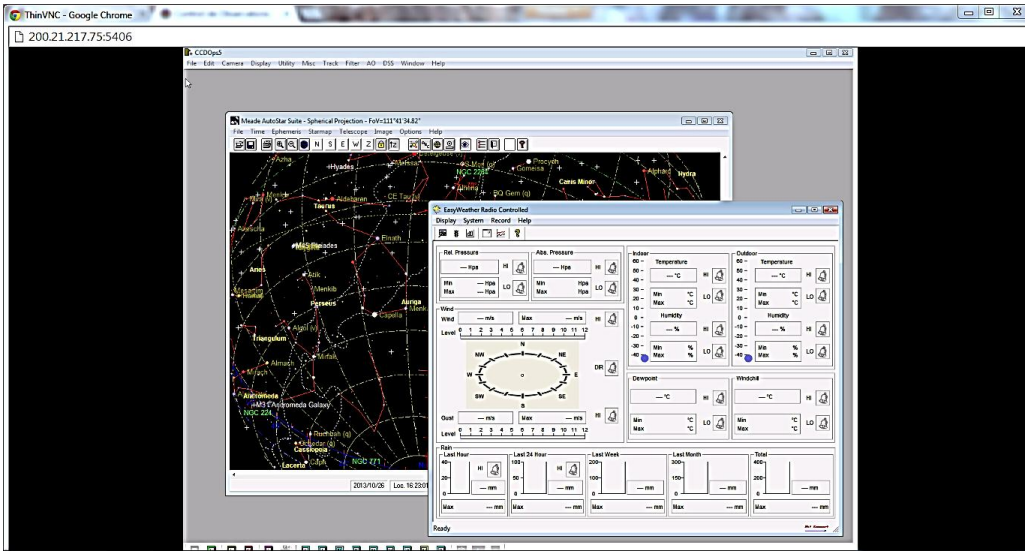


Figura 181. Página web con aplicaciones corriendo en VNC.

La aplicación EasyWeather entrega información como temperatura y humedad, con la información aportada por la estación meteorológica como se puede visualizar en la Figura 182.

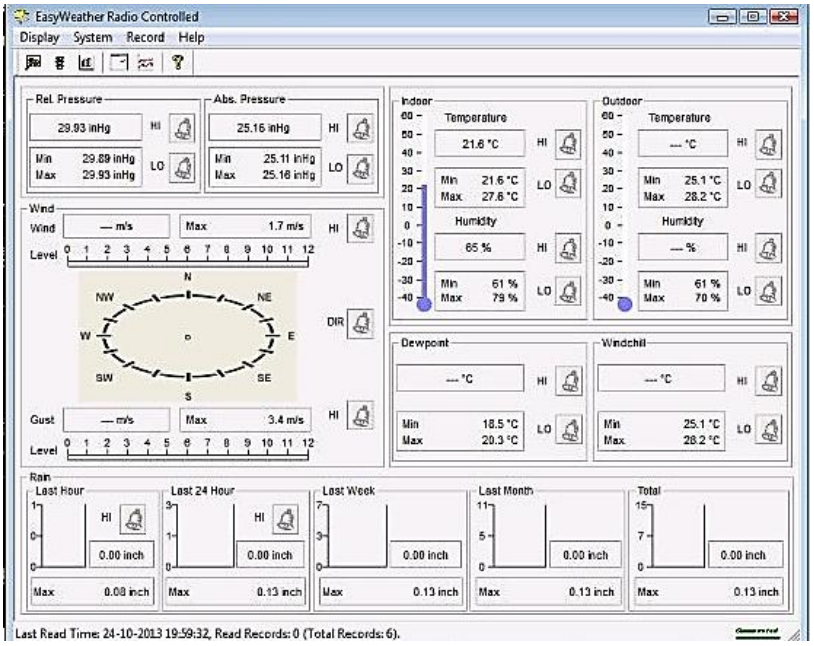


Figura 182. Aplicación EasyWeather corriendo sobre VNC

En la Figura 183 se observa la pestaña "Control Domo" donde se logra apreciar la existencia de cuatro iconos para controlar los elementos de hardware como la luminaria, los motores de la cúpula o de las compuertas.

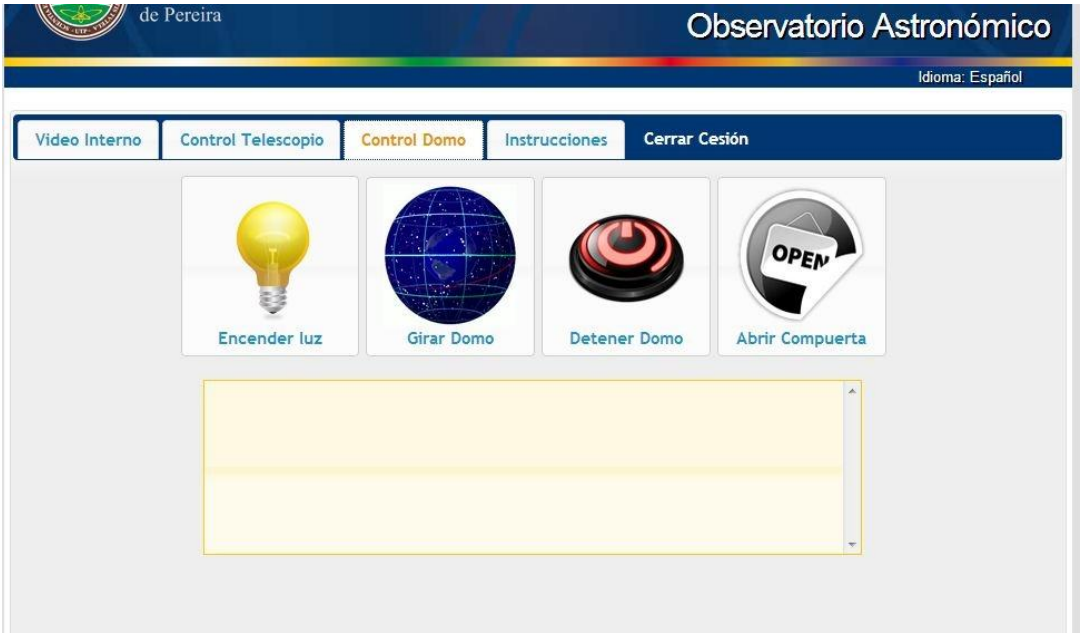


Figura 183. Pestaña “control domo”, con botones para controlar el hardware del Observatorio.

En la Figura 185 se observa el cambio de estado en el encendido y apagado de la luminaria ante cada evento de clic sobre el respectivo icono en la página web, de forma paralela se verifican los mensajes de las acciones realizadas en un cuadro de notificaciones, como se muestra en la Figura 184.

Figura 184. Acción de encendido y apagado de la luminaria.

Figura 185. Acción de encendido y apagado de la luminaria mediante control web.

El tiempo que se tarda entre el evento de hacer clic sobre el icono y ejecutar la acción sobre los elementos de hardware que realizan el control, es mínimo, pues las acciones resultan casi instantáneas, con retardo de tan solo algunos milisegundos.

Figura 186. Proceso de apertura de compuertas del domo mediante control web.

Posteriormente se verifica la apertura y cierre de las compuertas del domo, haciendo clic sobre los iconos "Abrir Compuerta" y "Cerrar compuerta".

De forma casi instantánea comienza la apertura de las compuertas y se logra apreciar en el cuadro de eventos que se visualiza en la Figura 186, los mensajes que indican el inicio y finalización de la operación, de igual manera se observa la apertura total de las compuertas y la parada automática cuando llega al final de la

posición indicado por los interruptores magnéticos, en la Figura 187 se aprecia la compuerta en estado de apertura finalizado.

Figura 187. Compuertas abiertas en su totalidad por acción de control vía web.

Posteriormente, se regresa a la página de control mediante VNC, se localiza la aplicación AutoStar y dentro de esta, la carta celeste como se visualiza en la Figura 188, se hace clic sobre el planeta Venus contenido en la carta celeste, solicitando de esta manera que el telescopio haga el movimiento necesario para apuntar hacia dicho planeta.

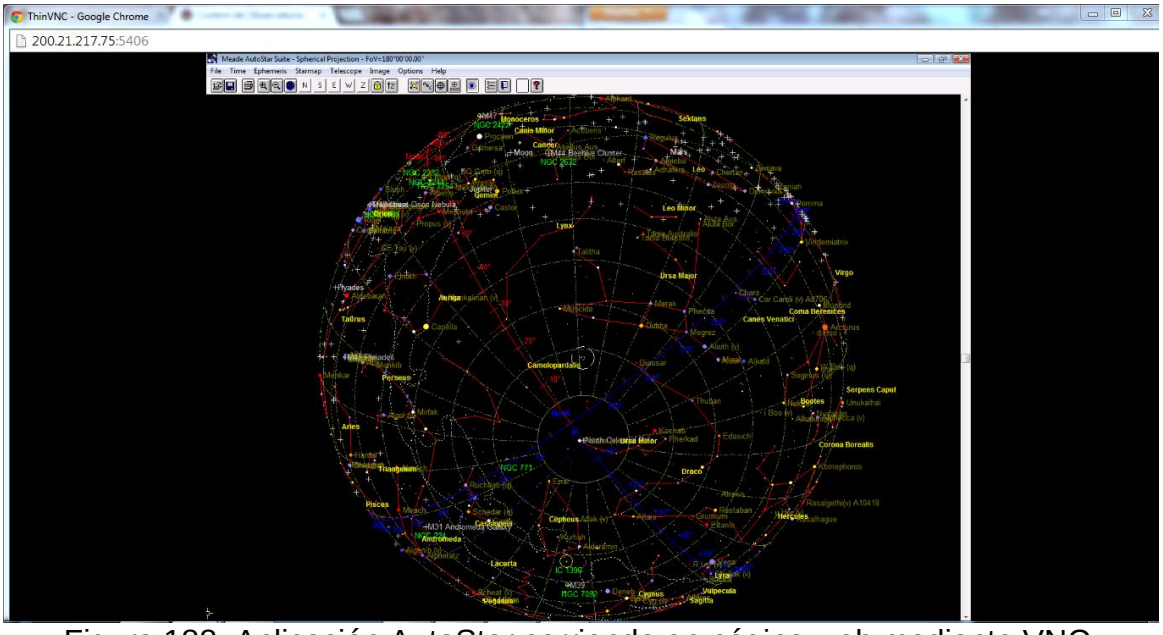


Figura 188. Aplicación AutoStar corriendo en página web mediante VNC.

El telescopio inicia el procedimiento de alineación con los movimientos necesarios para llegar hasta el punto indicado.

Posteriormente se hace clic sobre el botón "Girar Domo" de la Figura 186 y se verifica el correcto funcionamiento de control sobre el motor que permite el giro de la cúpula, se verifica además que el domo se detiene automáticamente cuando se encuentra alineado el telescopio y las compuertas del domo.

En la Figura 189 se evidencia cuando se verifica de forma paralela el contenido de la señal de video en streaming y la imagen obtenida por la cámara conectada al telescopio con la aplicación CCDOps corriendo en VNC.

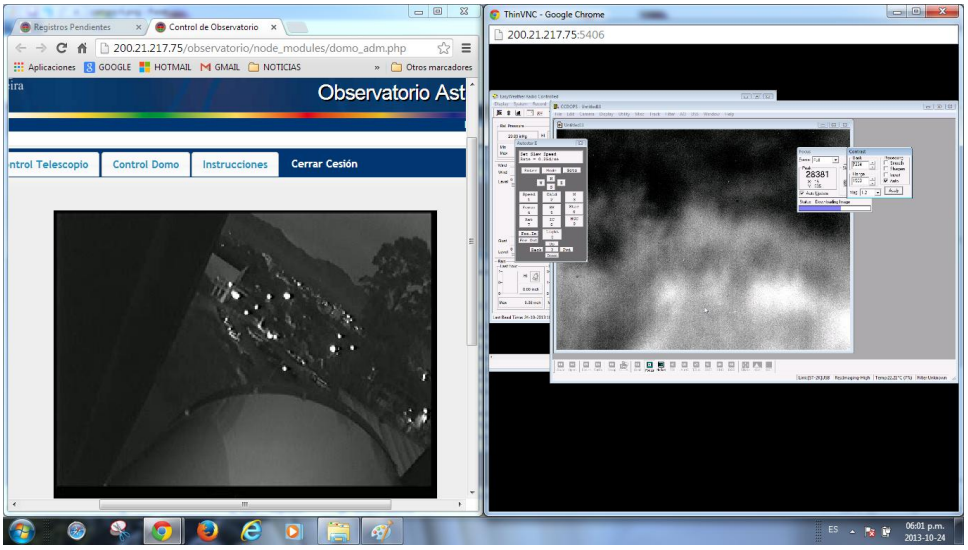


Figura 189. Visualización paralela de páginas web, lado izquierdo video en streaming, lado derecho aplicación CCDOps corriendo sobre VNC.

En la Figura 190 se observa como la aplicación CCDOps inicia el proceso de adquisición de la imagen desde el telescopio, obteniendo una imagen en vivo desde la plataforma web del planeta venus.

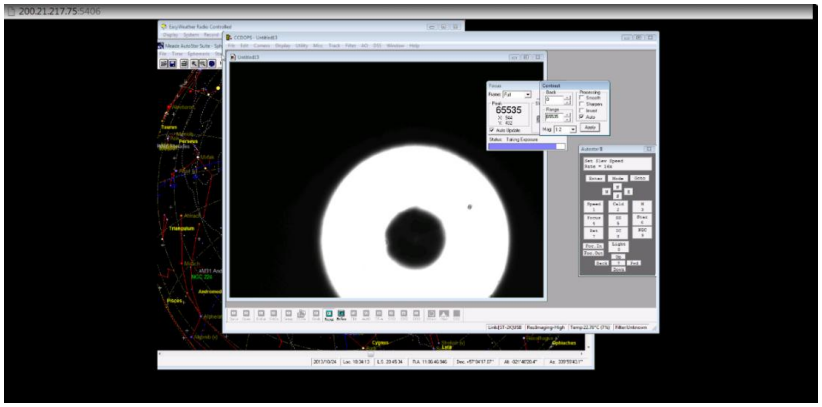


Figura 190. Aplicación CCDOps corriendo en página web sobre VNC.

Por último se hacen las configuraciones necesarias de alineación desde el control del AutoStar, para obtener una imagen más clara del planeta, como la visualizada en la Figura 191.

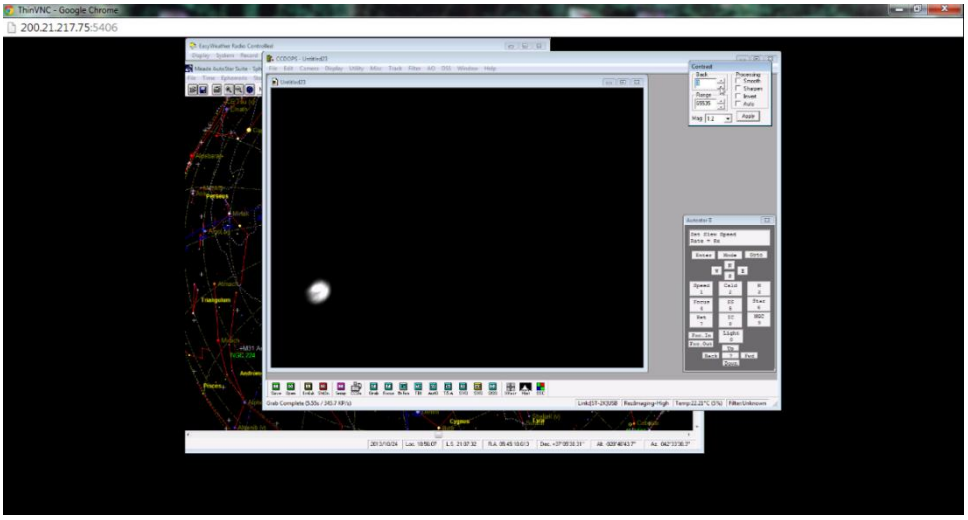


Figura 191. Imagen del planeta venus obtenida mediante control vía web del observatorio astronómico de la universidad Tecnológica de Pereira.

7

Conclusiones

Capítulo 7. Conclusiones

1. Actualmente la tecnología orientada a internet ha permitido la expansión y disponibilidad de aplicaciones desarrolladas en diferentes campos de acción, donde algunas requieren adicionalmente un desarrollo de hardware, para el caso del Observatorio Astronómico de la Universidad Tecnológica de Pereira, que solo contaba con controles manuales de los instrumentos ubicados allí, se logró por medio del diseño y construcción de un aplicativo web y una plataforma electrónica adecuada, el desarrollo de un sistema basado en software y hardware que conjuntamente permiten la disponibilidad y fácil acceso al control del Observatorio para cualquier usuario con acceso a internet y con los permisos y privilegios adecuados.

2. La necesidad de tener acceso a herramientas de investigación astronómica cada vez se hace más evidente, en la medida en que hay más estudiantes, profesores e investigadores interesados en conocer y estudiar el mundo de la astronomía, requiriendo herramientas tecnológicas que permitan globalizar no solo los contenidos de los estudios e investigaciones, sino también las herramientas de difícil acceso como lo es un observatorio astronómico. Mediante la implementación del sistema de control vía web del Observatorio Astronómico de la UTP se consiguen ventajas como el uso por parte de usuarios sin acceso a este tipo de herramientas tanto en Colombia como en cualquier parte del mundo, contribuyendo al desarrollo de la investigación científica y aportando un importante posicionamiento de la Universidad Tecnológica de Pereira por la implementación de este tipo de tecnologías para el desarrollo de la investigación astronómica.

3. Para la puesta en marcha del control vía web del Observatorio Astronómico, se hizo necesaria la unificación de diferentes módulos de hardware, combinados con diferentes tecnologías, aplicaciones y desarrollos de software, logrando una plataforma versátil con una integración adecuada de todos sus componentes, permitiendo tener un modelo estable y ágil del sistema tanto para el administrador de la plataforma como para sus usuarios.

Bibliografía

[1] WATANABE, k. OKADA, h. SAKAMOTO, m. TOYOMASU, s. OKYUDO,n. Classroom Lessons Of Astronomy Using The Internet And Its Multimedia Applications. IEEE International Conference: Multi Media Engineering Education, 1996.

[2] RIFKIN, A. Reengineering Hubble Space Telescope Control Center System. IEEE Internet Computing. Volume 1, 1997.

[3] DONG, Jian. WANG, Jian. JIN, Ge. DENG, Xiao-chao. YUAN, Hai-Long. Research of Application Layer for Large Astronomical Telescope Observatory Control System Framework. IEEE International Conference: Computational Intelligence and Software Engineering, 2009.

[4] LOPEZ, D. CEDAZO, R. SANCHEZ, F.M. SEBASTIAN, J.M. Ciclope Robot: Web-Based System to Remote Program an Embedded Real-Time System. IEEE Transactions: Industrial Electronics, Dec. 2009.

[5] CEDAZO, R. LOPEZ, D. SANCHEZ, F.M. SEBASTIAN, J.M. The Montegancedo Astronomical Observatory: The first free remote observatory for learning astronomy. Education Engineering: (EDUCON), 2010 IEEE.

[6] WATANABE, k. OTANI, m. TANAKA, h. OKYUDO, m. The Simple and Compact Remote Telescope System. International Conference: Complex, Intelligent and Software Intensive Systems (CISIS), 2010.

[7] Colaboradores Arduino. Instrucciones de manejo en la plataforma Arduino [en línea]. Arduino. Disponible en línea en: http://arduino.cc/en/Reference/HomePage.

[8] Colaboradores Atmel. Hoja de datos y especificaciones del Microcontrolador ATMEGA328P [en línea]. Atmel. Disponible en línea en: http://www.atmel.com/Images/8161s.pdf

[9] Andrés Oyarce, Paul Aguayo, Eduard Martin. Guía del Usuario XBEE Series 1. Ingeniería MCI LTDA.

[10] Colaboradores web electrónica. Teoría básica de transmisión en radio frecuencia. Buenas tareas. Disponible en línea: http://www.buenastareas.com/ensayos/Modulos-De-Transmision-y-Recepcion-Tipos-y/1714974.html.

[11] Josep Molina. Tecnología Reed. Apuntes. [en línea]. Disponible: http://www.apuntesdeelectronica.com/componentes/tecnologia-reed switch.htm.

[12] Colaboradores de DatasheetCatalog, L298. Hoja de datos [en línea]. Disponible:http://www.datasheetcatalog.com/datasheets_pdf/L/2/9/8/L298.s html.

[13] Colaboradores de la IEEE. Información actual de ese protocolo [en línea]. Disponible: http://www.ieee802.org/15/pub/TG4.html.

[14] Colaboradores de Sharp, Sensor Infrarrojo. Hoja de datos y especificación [en línea]. Disponible: http://www.sharpsma.com/webfm_send/1487.

[15] Colaboradores de Suconel, SSR. Hoja de datos y especificación [en línea]. Disponible:http://www.suconel.com/virtual/products-mainmenu-64.html?page=shop.product_details&flypage=flypage.tpl&product_id=6890930&category_id=93.

[16] Colaboradores de Arduino. Información de puerto serial [en línea]. Disponible: http://arduino.cc/es/Reference/SoftwareSerial.

[17] Colaboradores de Texas Instruments, Max232. Hoja de datos y especificación [en línea].Disponible: http://www.ti.com/lit/ds/symlink /max232 .pdf.

[18] Tim Converse, Joyce Park, Clark Morgan. Php5 y Mysql Bible. Wiley Publishing Inc.

[19] Colaboradores w3schools. Introducción al HTML [en línea]. W3schools. Disponible en línea en: http://www.w3schools.com/html/html_intro.asp.

[20] W. Jason Gilmore, Robert H. Treat. Comienzo del PHP y el PostgreSQL 8. Editorial Apress.

[21] Nicholas C. Zakas. JavaScript for Web Developers 2nd Edition. Editorial Wrox.

[22] Jonathan Chaffer, Karl Swedberg. Learning jQuery Third Edition. Editorial Packt Publishing Ltd.

[23] Colaboradores w3schools. Introducción al CCS [en línea]. W3schools. Disponible en línea en: http://www.w3schools.com/css/default.asp.

[24] Colaboradores de komova. Definiciones [en línea]. Disponibles: http://www.komova.net/video-streaming.html.

[25] Cybele Software. Guia de usuario ThinVNC HTML5 Remote Access. Editorial Cybele Software.

[26] Steven Roman, Ron Petrusha, Paul Lomax. VB .NET Language in a Nutshell. Primera edición agosto de 2011. Editorial O'Reilly.

[27] Colaboradores de msdn. Información de Lenguaje de ejecución común [en línea]. Disponible: http://msdn.microsoft.com/es-es/library/8bs2ecf4.aspx.

[28] Colaboradores de SerialPort. Información de instalación y uso [en línea]. Disponible: https://github.com/voodootikigod/node-serialport.

9

Anexos

Anexo 1. Código Microcontrolador 2

```
int sensor = A0;
int sensorValue;
char antArduino = 'A';

void setup() {
Serial.begin(9600);
pinMode(7,OUTPUT);
digitalWrite(7,LOW);
}

void loop() {
int Value = analogRead(A0);
char arduino = Serial.read();
int sensorValue = map(Value, 0, 1024, 0 , 200);
if(arduino){
if(arduino == 'A'){
digitalWrite(7,LOW);
antArduino = 'A';
}
if(arduino == 'E' || antArduino == 'E'){
digitalWrite(7,HIGH);
if(sensorValue >= 0  && sensorValue <= 5){
Serial.write("1");
}else if(sensorValue > 20){
Serial.write("0");
}
antArduino = 'E';
}
}
delay(100);
}
```

Anexo 2. Código Microcontrolador 1

```
#include <SoftwareSerial.h>

int  i = 1;
int  j = 0;
int  cuentaUnos = 0;
int  cuentaCeros = 0;
```

```
char bandera = 'J';
char data;
char dataXbee;
char antDataXbee = '0';
char banderaDomo = 'A';
SoftwareSerial mySerial(6, 7); // RX, TX

void setup()
{
Serial.begin(9600);

pinMode(11, INPUT);
pinMode(10, INPUT);
pinMode(9,OUTPUT);
pinMode(8,OUTPUT);
pinMode(12,OUTPUT);
pinMode(13,OUTPUT);

Serial.write("A");
mySerial.begin(9600);
}

void loop()
{
data     = mySerial.read();
dataXbee = Serial.read();

if(dataXbee == '0' || dataXbee == '1'){
Serial.println(dataXbee);
}
mySerial.println('H');
if (data){
if(digitalRead(10) == LOW){
if(data == 'F'){
digitalWrite(12,HIGH);
bandera = 'I';
i = 1;
}
if(data == 'C'){
mySerial.println("C");
}
if(bandera == 'K'){
digitalWrite(13,LOW);
bandera = 'J';
mySerial.println("C");
```

```
}
}
if(digitalRead(11) == LOW){
if(data == 'C'){
digitalWrite(13,HIGH);
bandera = 'K';
i = 1;
}
if(data == 'F'){
mySerial.println("F");
}
if(bandera == 'I'){
digitalWrite(12,LOW);
bandera = 'J';
mySerial.println("F");
}
}

if(data == 'D'){
if(banderaDomo == 'N'){
Serial.println("S");
delay(3000);
digitalWrite(9,LOW);
Serial.write("A");
mySerial.println('J');
Serial.flush();
}else{
digitalWrite(9,HIGH);
Serial.write("E");
mySerial.println('D');
}
}else if(data == 'J'){
Serial.println("S1");
banderaDomo = 'A';
Serial.write("A");
Serial.flush();
}else if(data == 'A'){
digitalWrite(8,HIGH);
mySerial.println('A');
}else if(data == 'B'){
digitalWrite(8,LOW);
mySerial.println('B');
}else if(data == 'G'){
digitalWrite(9,LOW);
mySerial.println('G');
```

```
Serial.write("A");
}else if(data == 'H'){
mySerial.println('H');
}
if(data == 'G'){
mySerial.println('G');
Serial.write("A");
}
}
if(dataXbee == '1' && antDataXbee == '1'){
if(data == 'D'){
Serial.println("S3");
banderaDomo = 'N';
}
i = 1;
}
if(dataXbee == '1' || dataXbee == '0'){
antDataXbee = dataXbee;
}

}
```

Anexo 3. Código SQL de Creación de Base de Datos “bd”

```
CREATE TABLE IF NOT EXISTS `log` (
  `id_usuario` int(11) NOT NULL,
  `hora_entrada` datetime NOT NULL,
  `hora_salida` datetime NOT NULL,
  PRIMARY KEY (`id_usuario`,`hora_entrada`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `solicitud_registro` (
  `id` int(20) NOT NULL,
  `nombre` varchar(40) CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT
NULL,
  `apellido` varchar(40) CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT
NULL,
  `email` varchar(40) CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT
NULL,
  `institucion` varchar(40) CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT
NULL,
  `pais` varchar(40) CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT NULL,
  `ciudad` varchar(40) CHARACTER SET utf8 COLLATE utf8_spanish_ci NOT
NULL,
```

```
  `comentarios` varchar(100) CHARACTER SET utf8 COLLATE utf8_spanish_ci
NOT NULL,
  `Password_usuario` varchar(256) CHARACTER SET utf8 COLLATE
utf8_spanish_ci NOT NULL,
  `registrado` int(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `tiempo_permitido` (
  `usuario_Cedula` int(11) NOT NULL,
  `hora_inicial` datetime NOT NULL,
  `hora_final` datetime NOT NULL,
  `logueado` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`usuario_Cedula`,`hora_inicial`,`hora_final`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `tiempo_solicitado` (
  `usuario_cedula` int(11) NOT NULL,
  `hora_inicial` datetime NOT NULL,
  `hora_final` datetime NOT NULL,
  `aprobada` int(1) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

CREATE TABLE IF NOT EXISTS `usuario` (
  `Cedula_usuario` varchar(11) NOT NULL,
  `Nombre_usuario` varchar(20) NOT NULL,
  `Apellido_usuario` varchar(20) NOT NULL,
  `Email_usuario` varchar(50) DEFAULT NULL,
  `Password_usuario` varchar(256) NOT NULL,
  `Tipo_usuario` varchar(20) NOT NULL DEFAULT 'USUARIO',
  `Institucion` varchar(80) NOT NULL,
  PRIMARY KEY (`Cedula_usuario`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Anexo 4. Código de creación de Sesión

```
<?PHP
date_default_timezone_set('America/Bogota');

include('config.php');

$user   = $_POST['Id'];
$pass   = $_POST['pass'];
```

```
if (empty($user) || empty($pass)){
echo json_encode(array('TEXT'=>'Debe digitar un usuario y contraseña
valido','VALIDA'=>'1','USER'=>'No'));
die("");
}

$checkuser            = mysql_query("SELECT Cedula_usuario FROM usuario
WHERE Cedula_usuario='" . $user . "'");
$username_exist = mysql_num_rows($checkuser);


if ($username_exist == "0"){
echo json_encode(array('TEXT'=>'Datos no validos','VALIDA'=>'1','USER'=>'No'));
die("");
}else{
$result         = mysql_query("SELECT count(*) c FROM usuario where
Cedula_usuario = " . $user . " and Password_usuario = md5('" . $pass . "')") or
die("No se puede consultar el usuario en la base de datos");
$num3       = mysql_fetch_array($result);

if (!$num3["c"]) {
echo json_encode(array('TEXT'=>'Contraseña
invalida','VALIDA'=>'1','USER'=>'No'));
die("");
}else{

$rs     = mysql_query("SELECT *  FROM usuario WHERE Cedula_usuario='" .
$user . "') or die("No se puede consultar el usuario en la base de datos");
$row   = mysql_fetch_assoc($rs);

// Start the login session
session_start();

$_SESSION['Id']                    = $user;
$_SESSION['nom']                   = $row['Nombre_usuario'];
$_SESSION['apellido']              = $row['Apellido_usuario'];
$_SESSION["ultimoAcceso"]      = date("Y-m-d H:i:s");

$hora_entrada                = $_SESSION["ultimoAcceso"];
$sql                         = "INSERT INTO log (id_usuario,hora_entrada)
VALUES (" . $user .",'" . $hora_entrada . "')";
mysql_query($sql) or die("Error en el registro de usuario. MySQL dice:" .
mysql_error());
```

```
if ($row['Tipo_usuario'] == 'ADMINISTRADOR') {
$_SESSION["autentificado"]        = "SIA";
echo json_encode(array('TEXT'=>'','VALIDA'=>'0','USER'=>'Ad'));
die("");
}else if($row['Tipo_usuario'] == 'USUARIO') {
$_SESSION["autentificado"]        = "SI";
echo json_encode(array('TEXT'=>'','VALIDA'=>'0','USER'=>'Us'));
die("");
}
}
}
?>
```

Anexo 5. Código Solicitud Acceso a Plataforma

```
<?PHP
require("css/cabezote.php");
?>
<script>
$(document).ready(function(){
document.title = "Registro";
$("div.utp-menu-nav-subtitulo").html("<h2>Solicitud registro de usuario</h2>");

$('#login').click(function(){
window.location = 'index.php';
});

});
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<script src="js/jquery-ui-1.10.3.custom/js/jquery-ui-1.10.3.custom.js"></script>
<link rel="stylesheet" href="js/jquery-ui-1.10.3.custom/css/ui-lightness/jquery-ui-
1.10.3.custom.css"></link>
<div id="utp-content-cont" class="grid_16 prefix_1 omega">
<div class="utp-menu-nav-subtitulo">
</div>                                                    <!-- Titulo del contenido -->
</div>
<div class="clear"></div>
<div align="center" id="content">
<h1 class="formulario">
<div>
<span class="show">Cédula</span><br>
<INPUT TYPE="TEXT" NAME="id" id="cedula" class="texto">
</div>
```

```
<div>
<span class="show">Nombre</span><br>
<INPUT TYPE="TEXT" NAME="nombre" id="nombre" class="texto">
</div>
<div>
<span class="show">Apellido</span><br>
<INPUT TYPE="TEXT" NAME="apellido" id="apellido" class="texto">
</div>
<div>
<span class="show">E-mail</span><br>
<INPUT TYPE="TEXT" NAME="email" id="email" class="texto">
</div>
<div>
<span class="show">Contraseña</span><br>
<INPUT TYPE="password" id="pass" NAME="pass" class="texto">
</div>
<div>
<span class="show">Repetir contraseña</span><br>
<INPUT TYPE="password" id="pass2" NAME="pass2" class="texto">
</div>
<div>
<span class="show">Institución</span><br>
<INPUT TYPE="TEXT" id="institucion" NAME="institucion">
</div>
<div>
<span class="show">Pais</span><br>
<INPUT TYPE="TEXT" id="pais" NAME="pais">
</div>
<div>
<span class="show">Ciudad</span><br>
<INPUT TYPE="TEXT" id="ciudad" NAME="ciudad">
</div>
<div>
<span class="show">Objetivo de Uso</span><br>
<textarea name="comentarios" id="coment" rows="10" cols="40"></textarea>
</div>
<br>
<INPUT TYPE="BUTTON" id="insertar" value="Enviar">
<INPUT TYPE="BUTTON" id="login" value="Login">
</h1>
</div>
<div id="dialog"></div>
<script>
var exitos = false;
$('#insertar').click(function(){
```

```
$.ajax({
data    :{      id                      : $('#cedula').val(),
nombre                  : $('#nombre').val(),
apellido        : $('#apellido').val(),
pais            : $('#pais').val(),
ciudad          : $('#ciudad').val(),
coment                  : $('#coment').val(),
pass            : $('#pass').val(),
pass2           : $('#pass2').val(),
email           : $('#email').val(),
institucion : $('#institucion').val()
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :'solicitud_registro_2.php',
success         :function(data){
var dataObj = eval('(' + data + ')');
if(dataObj.VALIDA == '1'){
$('#dialog').html('<p>' + dataObj.TEXT + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});
}else{
$('#dialog').html('<p>' + dataObj.TEXT + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
```

```
}
});
exitos = true;
}

$('span.ui-icon-closethick').click(function(){

if(exitos){
window.location = 'solicitud_registro.php';
}
});
},
beforeSend:function(data){$('div.utp-menu-nav-subtitulo').html('<h2>Solicitud
Registro</h2>')},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
});


</script>
<?php
require("css/piezote.php");
?>
```

Anexo 6. Código de Perfil de Usuario

```
<?PHP
include('config.php');

$id                     =       $_POST["id"];
$nombre                 =       $_POST["nombre"];
$apellido               =       $_POST["apellido"];
$email                  =       $_POST["email"];
$institucion            =       $_POST["institucion"];
$pais                   =       $_POST["pais"];
$ciudad                 =       $_POST["ciudad"];
$comentarios            =       $_POST["coment"];
$checkuser              =       mysql_query("SELECT Cedula_usuario FROM
usuario WHERE Cedula_usuario ='" . $id . "'");
$username_exist         =       mysql_num_rows($checkuser);
$pass                   =       md5(trim($_POST["pass"]));
$pass2                  =       md5(trim($_POST["pass2"]));

if ($username_exist > 0){
```

```
echo json_encode(array('TEXT'=> "Este número de identificacion ya se encuentra
Registrado",'VALIDA'=>'1'));
die ("");
}

if($_POST["pass"] != $_POST["pass2"]){
echo json_encode(array('TEXT'=>"Las contraseñas no son iguales, por favor
intentelo nuevamente",'VALIDA'=>'1'));
die ("");
}

if(empty($nombre) || empty($id) || empty($apellido) || empty ($email) ||
empty($institucion) || empty($pais) || empty($ciudad) || empty($comentarios) ||
empty($_POST["pass"]) || empty($_POST["pass2"])){
echo json_encode(array('TEXT'=>"Datos incompletos, por favor intentelo
nuevamente",'VALIDA'=>'1'));
die ("");
}

if($_POST['email'] == '' or !preg_match("/^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-
Z0-9\-\.]+$/",$_POST['email'])){
echo json_encode(array('TEXT'=>"Por favor ingrese un Email
correcto",'VALIDA'=>'1'));
die ("");
}

if($_POST['id'] == '' or !preg_match("/^[0-9]/",$_POST['id'])){
echo json_encode(array('TEXT'=>"Por favor ingrese un numero de identificacion
válido",'VALIDA'=>'1'));
die ("");
}
/* Ejecución de la sentencia SQL INSERT*/
$sql    = "INSERT INTO
solicitud_registro(id,nombre,apellido,email,institucion,pais,ciudad,comentarios,
Password_usuario) VALUES ('". $id ."','" .$nombre ."','" . $apellido ."','" . $email .
"','" . $institucion . "', '" . $pais ."', '" . $ciudad ."', '" . $comentarios . "', '" . $pass ."')";
mysql_query($sql) or die( json_encode(array('TEXT'=>"Ya se encuentra una
solicitud de registro pendiente",'VALIDA'=>'1')) ); //MySQL dice:".mysql_error()

echo json_encode(array('TEXT'=>"Solicitud enviada correctamente, esta
información será validada por el administrador en un término menor a tres días
hábiles, posteriormente recibirá un correo informando sobre el estado de su
solicitud. Guarde su contraseña en un lugar seguro para poder ingresar luego de
ser aprobado su registro.",'VALIDA'=>'0'));
```

```
/*ENVIO DE E-MAIL A ADMINISTRADOR DE LA CUENTA*/
mail('jefferson88ve@gmail.com',"Solicitud de registro en Observatorio UTP"," " .
$nombre . " " .$apellido . " de la institución " . $institucion . " ha solicitado el
registro en la plataforma del Observatorio Astronómico de la UTP. " . $pais . ": " .
$ciudad . "");
?>
```

Anexo 7. Código Reporte de Últimos Accesos

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
document.title = "Ultimos Accesos";
});
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<div align="center" id="content" >
<h1 class="tabla">
<?php
include('config.php');
$user       = $_SESSION['Id'] ;
$result     = mysql_query("SELECT * FROM usuario u LEFT JOIN log t on
(u.cedula_Usuario=t.id_usuario) WHERE NOT tipo_usuario ='ADMINISTRADOR'
AND Cedula_usuario=$user ORDER BY Apellido_usuario,Nombre_usuario");
?>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1"
frame="void" bordercolor="#F3F3F3">
<thead class="style3">
<tr>
<td bgcolor="#2b57ad" align="center"><span class="show">Nombre</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Apellido</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Hora
inicial</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Hora
final</span></td>
</tr>
</thead>
```

```
<tbody>
<?php
/* -- Mostramos los registros -- */
while ($row = @mysql_fetch_array($result)) {
echo '<tr class="style6">
<td  bgcolor="#FFFFFF" height="20"><span class="show">' .
$row["Nombre_usuario"] . '</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] .
'</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_entrada"] .
'</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_salida"] .
'</span></td>
</tr>';
}
@mysql_free_result($result);
?>
</tbody>
</table>
</h1>
</div>
```

Anexo 8. Código Solicitud Acceso a Control Vía web

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<link rel="stylesheet" media="all" type="text/css"
href="http://code.jquery.com/ui/1.8.23/themes/smoothness/jquery-ui.css" />
<link rel="stylesheet" media="all" type="text/css" href="datetimepicker/css/jquery-
ui-timepicker-addon.css" />
<link rel="stylesheet" media="all" type="text/css"
href="datetimepicker/css/time.css" />
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="datetimepicker/jquery-1.8.2.min.js"></script>
<script type="text/javascript" src="datetimepicker/jquery-ui.min.js"></script>
<script type="text/javascript" src="datetimepicker/jquery-ui-timepicker-
addon.js"></script>
```

Anexos

```
<script type="text/javascript" src="datetimepicker/jquery-ui-
sliderAccess.js"></script>
<script>
$(document).ready(function(){
document.title = "Solicitud";
});
</script>
<div align="center" id="content" >
<h1 class="formulario">
<?php
include('config.php');
echo '<div>
<span class="show"> Nombre Usuario</span><br>';

$id         = $_SESSION["Id"];

/* -- Creamos la sentencia SQL y la ejecutamos -- */
$sSQL       = "SELECT Cedula_usuario,Nombre_usuario,Apellido_usuario  FROM
usuario WHERE Cedula_usuario='$id'";
$result     = mysql_query($sSQL);

echo '<select id="cedula" name="cedula">';

/* -- Generamos el menu desplegable -- */
while ($row = mysql_fetch_array($result)) {
echo '<option value=' . $row["Cedula_usuario"] . '>' . $row["Nombre_usuario"] . ' ' .
$row["Apellido_usuario"] . '</option>';
}
echo '</select>
<br><br>
</div>';
?>
<div>
<span class="show">Hora inicial</span><br>
<!--<INPUT TYPE="TEXT" NAME="id" id="cedula" class="texto"> -->
<input type="text" size="25" name="t1" id="1" style="z-index:10000;" value="" />
</div>
<div>
<span class="show">Hora Final</span><br>
<!--<INPUT TYPE="TEXT" NAME="id" id="cedula" class="texto"> -->
<input type="text" size="25" name="t2" id="2" style="z-index:10000;" value="" />
</div>
<br>
<input type="submit" id="solicitar" value="Solicitar">
</h1>
```

```
</div>
<div id="dialog"></div>
<script>
$().ready(function(){
$('#solicitar').click(function(){
//2KQT8-HV27P-GTTV9-2WBVV-M7X96 SERIAL VISUAL STUDIO
http://www.youtube.com/watch?v=q-llprAOO54
$.ajax({
data    :{ cedula: $('#cedula').val(),
t1:      $('#1').val(),
t2:  $('#2').val()
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url                :'solicitud2.php',
success        :function(data){
$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});
},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
});

var show = $('#1').datetimepicker({
dateFormat: "yy'-'mm'-'dd",
separator: ' ',
ampm: false,
onSelect:function(data){
$('#1').attr('value',data);
}
});
```

```
console.dir(show);

var show2 = $('#2').datetimepicker({
dateFormat: "yy'-'mm'-'dd",
separator: ' ',
ampm: false,
onSelect:function(data){
$('#2').attr('value',data);
}
});

console.dir(show2);
});
</script>


<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<?php
include('config.php');

$cedula = $_POST["cedula"];
$t1 = $_POST["t1"];
$t2 = $_POST["t2"];

/* -- Creamos la sentencia SQL - SELECT y la ejecutamos -- */
$sSQL       = "SELECT Cedula_usuario,Nombre_usuario,Apellido_usuario FROM
usuario WHERE Cedula_usuario='$cedula'";
$result     = mysql_query($sSQL);

while ($row = mysql_fetch_row($result)){
$nombre         = $row[1];
$apellido       = $row[2];
}
/* --Ejecucion de la sentencia SQL  -- */
//$result = "select count(*) c from tiempo_permitido where (hora_inicial between
'$t1' and '$t2') or (hora_final between '$t1' and '$t2')";
```

```
$result     = mysql_query("SELECT count(*) c FROM tiempo_permitido WHERE
(hora_inicial between '$t1' and '$t2') OR (hora_final between '$t1' and '$t2')") or
die("Couldn't query the user-database.");
$num        = mysql_fetch_array($result);
if(empty($t1) || empty($t2) || empty($cedula)){
echo "No se registraron valores verifique nuevamente.";
die("");
}
if ("$t1" >= "$t2") {
echo "El horario final no puede ser menor o igual al inicial.";
die("");
}else{
if ($num["c"]>0) {
echo "Ya existe un usuario autorizado en este rango de tiempo '.$t1.' - '.$t2.'.";
die("");
} else {
$sql = "INSERT INTO tiempo_solicitado(usuario_Cedula,hora_inicial,hora_final)
VALUES ($cedula,'$t1','$t2')";
mysql_query($sql) or die("");
echo "Solicitud enviada correctamente";
mail('jefferson88ve@gmail.com',"Solicitud de ingreso a Observatorio UTP","El
señor(a), $nombre $apellido ha solicitado el ingreso al Observatorio Astronómico
de la UTP desde $t1 hasta $t2");

}
}
?>
```

Anexo 9. Código de Reporte de Horarios no Disponibles para el Acceso

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
document.title = "Horarios Utilizados";
});
```

```
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<div align="center" id="content" >
<h1 class="tabla">
<?php
include('config.php');
$user       = $_SESSION['Id'] ;
$result     = mysql_query("select * from usuario u left join tiempo_solicitado t on
(u.cedula_Usuario=t.usuario_cedula) where Cedula_usuario!='' and hora_inicial
order by Apellido_usuario,Nombre_usuario");
?>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1"
frame="void" bordercolor="#F3F3F3">
<thead class="style3">
<tr>
<td bgcolor="#2b57ad" align="center"><span class="show"> Nombre</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Apellido</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Hora
inicial</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Hora
final</span></td>
</tr>
</thead>
<tbody>
<?php
/* -- Mostramos los registros -- */
while ($row = @mysql_fetch_array($result)) {
echo '<tr class="style6">
<td  bgcolor="#FFFFFF" height="20"><span class="show">' .
$row["Nombre_usuario"] . '</span></td>
<td  bgcolor="#FFFFFF" height="20"><span class="show">' .
$row["Apellido_usuario"] . '</span></td>
<td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["hora_inicial"] .
'</span></td>
<td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["hora_final"] .
'</span></td>
</tr>';
}
@mysql_free_result($result);
?>
</tbody>
</table>
</h1>
</div>
```

Anexo 10. Código Para la Edición de Datos Personales

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
document.title = "Editar Datos";
});
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" href="js/jquery-ui-1.10.3.custom/css/ui-lightness/jquery-ui-
1.10.3.custom.css"></link>
<script src="js/jquery-ui-1.10.3.custom/js/jquery-ui-1.10.3.custom.js"></script>
<div align="center" id="content" >
<h1 class="formulario">
<?php
include('config.php');

echo "<FORM METHOD=\"POST\" ACTION=\"#\"id=\"modificar\">";
include('config.php');

$id         = $_SESSION["Id"];
//Creamos la sentencia SQL y la ejecutamos
$sSQL       = "SELECT * FROM usuario WHERE Cedula_usuario='$id'";
$result     = mysql_query($sSQL);
$row        = mysql_fetch_array($result);
$cedula     = $row["Cedula_usuario"];
$nombre     = $row["Nombre_usuario"];
$apellido   = $row["Apellido_usuario"];
$email      = $row["Email_usuario"];
$pass       = $row["Password_usuario"];
echo'
<div>
<span class="show">Cédula </span><br>
<INPUT TYPE="TEXT" NAME="id" id="cedula" readonly="readonly" value="' .
$cedula . '" class="texto">
```

```
</div>
<div>
<span class="show">Nombre</span><br>
<INPUT TYPE="TEXT" NAME="nombre" id="nombre" value="' . $nombre . '"
class="texto">
</div>
<div>
<span class="show">Apellido</span><br>
<INPUT TYPE="TEXT" NAME="apellido" id="apellido" value="' . $apellido . '"
class="texto">
</div>
<div>
<span class="show">Contraseña</span><br>
<INPUT TYPE="password" id="pass" NAME="pass" class="texto">
</div>
<div>
<span class="show">Repetir contraseña</span><br>
<INPUT TYPE="password" id="pass2" NAME="pass2" class="texto">
</div>
<div>
<span class="show">E-mail</span><br>
<INPUT TYPE="TEXT" NAME="email" id="email" value="' . $email . '"
class="texto">
</div>
<br>
<INPUT TYPE="button" id="Modificar" value="Modificar">
';
?>
</FORM>
</h1>
</div>
<div id="dialog"></div>
<script>

$('#Modificar').click(function(){
$.ajax({
data   :{       cedula          : $('#cedula').val(),
nombre                  : $('#nombre').val(),
apellido        : $('#apellido').val(),
pass            : $('#pass').val(),
pass2           : $('#pass2').val(),
email           : $('#email').val(),
},
type    :"POST",
dataType:"html",
```

```
async :false,
cache :false,
url              :'editar_datos3.php',
success       :function(data){

$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});

},
beforeSend:function(data){$('div.utp-menu-nav-subtitulo').html('<h2>Editar
Usuario</h2>')},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
});
</script>


<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<?php
include('config.php');

$nombre        = $_POST["nombre"];
$cedula        = $_POST["cedula"];
$apellido      = $_POST["apellido"];
$email         = $_POST["email"];
$pass          = md5(trim($_POST["pass"]));
$pass2         = md5(trim($_POST["pass2"]));
```

```
if ($_POST['pass'] != $_POST['pass2']) {
echo "Las contraseñas no son iguales, por favor intentelo nuevamente";
die("");
}

if (empty($nombre) || empty($cedula) || empty($_POST['pass']) ||
empty($_POST['pass2']) || empty($apellido) || empty($email)) {
echo "Datos incompletos, por favor intentelo nuevamente '.$cedula.'";
die("");
}

if($_POST['email'] == '' or !preg_match("/^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-
Z0-9\-\.]+$/",$_POST['email']))
{
echo "Por favor ingrese un Email correcto";
die ("");
}

/* -- Creamos la sentencia SQL y la ejecutamos - UPDATE -- */
$sSQL = "UPDATE usuario SET Nombre_usuario ='$nombre',Apellido_usuario
='$apellido',Email_usuario ='$email',Password_usuario ='$pass' WHERE
Cedula_usuario ='$cedula'";
mysql_query($sSQL) or die("Error en la creacion. MySQL dice:" . mysql_error());
echo "Datos modificados correctamente";
?>
```

Anexo 11. Código de Solicitud de Control de Plataforma

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<link rel="stylesheet" media="all" type="text/css"
href="http://code.jquery.com/ui/1.8.23/themes/smoothness/jquery-ui.css" />
<link rel="stylesheet" media="all" type="text/css" href="datetimepicker/css/jquery-
ui-timepicker-addon.css" />
<link rel="stylesheet" media="all" type="text/css"
href="datetimepicker/css/time.css" />
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
```

```
<script type="text/javascript" src="datetimepicker/jquery-1.8.2.min.js"></script>
<script type="text/javascript" src="datetimepicker/jquery-ui.min.js"></script>
<script type="text/javascript" src="datetimepicker/jquery-ui-timepicker-
addon.js"></script>
<script type="text/javascript" src="datetimepicker/jquery-ui-
sliderAccess.js"></script>
<script>
$(document).ready(function(){
document.title = "Solicitud";
});
</script>
<div align="center" id="content" >
<h1 class="formulario">
<?php
include('config.php');
echo '<div>
<span class="show"> Nombre Usuario</span><br>';

$id         = $_SESSION["Id"];

/* -- Creamos la sentencia SQL y la ejecutamos -- */
$sSQL       = "SELECT Cedula_usuario,Nombre_usuario,Apellido_usuario  FROM
usuario WHERE Cedula_usuario='$id'";
$result     = mysql_query($sSQL);

echo '<select id="cedula" name="cedula">';

/* -- Generamos el menu desplegable -- */
while ($row = mysql_fetch_array($result)) {
echo '<option value=' . $row["Cedula_usuario"] . '>' . $row["Nombre_usuario"] . ' ' .
$row["Apellido_usuario"] . '</option>';
}
echo '</select>
<br><br>
</div>';
?>
<div>
<span class="show">Hora inicial</span><br>
<!--<INPUT TYPE="TEXT" NAME="id" id="cedula" class="texto"> -->
<input type="text" size="25" name="t1" id="1" style="z-index:10000;" value="" />
</div>
<div>
<span class="show">Hora Final</span><br>
<!--<INPUT TYPE="TEXT" NAME="id" id="cedula" class="texto"> -->
<input type="text" size="25" name="t2" id="2" style="z-index:10000;" value="" />
```

```
</div>
<br>
<input type="submit" id="solicitar" value="Solicitar">
</h1>
</div>
<div id="dialog"></div>
<script>
$().ready(function(){
$('#solicitar').click(function(){
//2KQT8-HV27P-GTTV9-2WBVV-M7X96 SERIAL VISUAL STUDIO
http://www.youtube.com/watch?v=q-llprAOO54
$.ajax({
data    :{ cedula: $('#cedula').val(),
t1:      $('#1').val(),
t2:  $('#2').val()
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :'solicitud2.php',
success         :function(data){
$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});
},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
});

var show = $('#1').datetimepicker({
dateFormat: "yy'-'mm'-'dd",
separator: ' ',
ampm: false,
onSelect:function(data){
```

```
$('#1').attr('value',data);
}
});

console.dir(show);

var show2 = $('#2').datetimepicker({
dateFormat: "yy'-'mm'-'dd",
separator: ' ',
ampm: false,
onSelect:function(data){
$('#2').attr('value',data);
}
});

console.dir(show2);
});
</script>


<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<?php
include('config.php');

$cedula = $_POST["cedula"];
$t1 = $_POST["t1"];
$t2 = $_POST["t2"];

/* -- Creamos la sentencia SQL - SELECT y la ejecutamos -- */
$sSQL       = "SELECT Cedula_usuario,Nombre_usuario,Apellido_usuario FROM
usuario WHERE Cedula_usuario='$cedula'";
$result     = mysql_query($sSQL);

while ($row = mysql_fetch_row($result)){
$nombre         = $row[1];
$apellido       = $row[2];
}
/* --Ejecucion de la sentencia SQL  -- */
```

```
//$result = "select count(*) c from tiempo_permitido where (hora_inicial between
'$t1' and '$t2') or (hora_final between '$t1' and '$t2')";

$result     = mysql_query("SELECT count(*) c FROM tiempo_permitido WHERE
(hora_inicial between '$t1' and '$t2') OR (hora_final between '$t1' and '$t2')") or
die("Couldn't query the user-database.");
$num        = mysql_fetch_array($result);
if(empty($t1) || empty($t2) || empty($cedula)){
echo "No se registraron valores verifique nuevamente.";
die("");
}
if ("$t1" >= "$t2") {
echo "El horario final no puede ser menor o igual al inicial.";
die("");
}else{
if ($num["c"]>0) {
echo "Ya existe un usuario autorizado en este rango de tiempo '.$t1.' - '.$t2.'.";
die("");
} else {
$sql = "INSERT INTO tiempo_solicitado(usuario_Cedula,hora_inicial,hora_final)
VALUES ($cedula,'$t1','$t2')";
mysql_query($sql) or die("");
echo "Solicitud enviada correctamente";
mail('jefferson88ve@gmail.com',"Solicitud de ingreso a Observatorio UTP","El
señor(a), $nombre $apellido ha solicitado el ingreso al Observatorio Astronómico
de la UTP desde $t1 hasta $t2");

}
}
?>
```

Anexo 12. Código Reporte de Permisos Asignados

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SIA") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
```

```
document.title = "Consulta de Usuario";
});
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<div align="center" id="content">
<h1 class="tabla">
<?PHP
include('config.php');
$result = mysql_query("SELECT * FROM usuario u LEFT JOIN tiempo_permitido t
ON (u.cedula_Usuario=t.usuario_Cedula) WHERE NOT tipo_usuario
='ADMINISTRADOR' AND hora_inicial!='' ORDER BY hora_inicial
DESC,hora_final DESC,Apellido_usuario,Nombre_usuario LIMIT 20");
?>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1"
frame="void" bordercolor="#F3F3F3">
<thead class="style3">
<tr>
<td bgcolor="#2b57ad" align="center"><span class="show">Cédula</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Nombre</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Apellido</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Hora
inicial</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Hora
final</span></td>
</td>
</thead>
<tbody>
<?php
/* -- Se muestra los registros --*/
while ($row = @mysql_fetch_array($result)) {

echo '<tr class="style6">
<td bgcolor="#FFFFFF" height="20"><span class="show">' .
$row["Cedula_usuario"] . '</span></td>
<td bgcolor="#FFFFFF"><span class="show">' . $row["Nombre_usuario"] .
'</span></td>
<td bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] .
'</span></td>
<td bgcolor="#FFFFFF"><span class="show">' . $row["hora_inicial"] .
'</span></td>
<td bgcolor="#FFFFFF"><span class="show">' . $row["hora_final"] . '</span></td>
</tr>';
}
@mysql_free_result($result);
?>
```

```
</tbody>
</table>
</h1>
</div>
```

Anexo 13. Código Reporte de Actividades

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SIA") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
document.title = "Reporte de Actividad";
});
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<div align="center" id="content">
<h1 class="tabla">
<?php
include('config.php');
$user   = $_SESSION['Id'] ;
$result = mysql_query("SELECT * FROM usuario u LEFT JOIN log t ON
(u.cedula_Usuario=t.id_usuario) WHERE tipo_usuario ='ADMINISTRADOR' AND
hora_salida!='0000-00-00 00:00:00' OR tipo_usuario='USUARIO' AND
hora_salida!='0000-00-00 00:00:00' ORDER BY hora_entrada DESC,hora_salida
DESC,Apellido_usuario,Nombre_usuario LIMIT 20");
?>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1"
frame="void" bordercolor="#F3F3F3">
<thead class="style3">
<tr>
<td bgcolor="#2b57ad" align="center"><span class="show">Cedula</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Nombre</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Apellido</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Hora
inicial</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">Hora
final</span></td>
```

```
</tr>
</thead>
<tbody>
<?php
/* -- Mostramos los registros -- */
while ($row = @mysql_fetch_array($result)) {
echo '<tr class="style6">
<td  bgcolor="#FFFFFF" height="20"><span class="show">' .
$row["Cedula_usuario"] . '</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["Nombre_usuario"] .
'</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] .
'</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_entrada"] .
'</span></td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_salida"] .
'</span></td>
</tr>';
}
@mysql_free_result($result);
?>
</tbody>
</table>
</h1>
</div>
```

Anexo 14. Código Reporte de Solicitudes Pendientes

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SIA") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
document.title = "Solicitudes Pendientes";
});
</script>
<!-- <script language="javascript" src="js/jquery-1.8.2.min.js"></script> -->
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
```

```
<link rel="stylesheet" href="js/jquery-ui-1.10.3.custom/css/ui-lightness/jquery-ui-
1.10.3.custom.css"></link>
<script src="js/jquery-ui-1.10.3.custom/js/jquery-ui-1.10.3.custom.js"></script>
<script language="javascript">
function desaprobar(posicion){
$.ajax({
data   :{  bandera            : "0",
cedula          : $('#cedula' + posicion).val(),
nombre                  : $('#nombre' + posicion).val(),
apellido        : $('#apellido' + posicion).val(),
hora_inicial: $('#hora_inicial' + posicion).val(),
hora_final     : $('#hora_final' + posicion).val()
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"solicitudes_pendientes_2.php",
success         :function(data){
$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});
againTableTwo();
},
beforeSend:function(data){$("div.utp-menu-nav-subtitulo").html("<h2>Solicitudes
Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

function againTableTwo(){
$.ajax({
data   :{  bandera            : "0"
},
```

```
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"solicitudes_pendientes.php",
success         :function(data){$("div.content").html(data);},
beforeSend:function(data){$("div.content").html("<img src=\"\">"); $("div.utp-menu-
nav-subtitulo").html("<h2>Solicitudes Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

function aprobado(posicion){
$.ajax({
data    :{  bandera             : "1",
cedula          : $('#cedula' + posicion).val(),
nombre                  : $('#nombre' + posicion).val(),
apellido        : $('#apellido' + posicion).val(),
hora_inicial: $('#hora_inicial' + posicion).val(),
hora_final      : $('#hora_final' + posicion).val()
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"solicitudes_pendientes_2.php",
success         :function(data){
$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});
againTable();
},
beforeSend:function(data){$("div.utp-menu-nav-subtitulo").html("<h2>Solicitudes
Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
```

```
});
}

function againTable(){
$.ajax({
data    :{  bandera             : "0"},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"solicitudes_pendientes.php",
success         :function(data){$("div.content").html(data);},
beforeSend:function(data){$("div.content").html("<img src=\"\">"); $("div.utp-menu-
nav-subtitulo").html("<h2>Solicitudes Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

</script>
<div id="dialog"></div>
<div align="center" id="content">
<h1 class="tabla">
<?php
include('config.php');

$user   = $_SESSION['Id'];
$result = mysql_query("SELECT * FROM usuario u LEFT JOIN tiempo_solicitado t
ON (u.cedula_Usuario=t.usuario_cedula) WHERE Tipo_usuario='USUARIO'
ORDER BY Apellido_usuario,Nombre_usuario");
?>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1"
frame="void" bordercolor="#F3F3F3">
<thead class="style3">
<tr>
<td bgcolor="#2b57ad" align="center"><span class="show"> Cédula</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Nombre</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Apellido</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Hora
inicial</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Hora
final</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Estado</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Asignar</span></td>
</tr>
</thead>
```

```
<tbody>
<?php
/* -- Mostramos los registros -- */
$i  = 0;
while ($row = @mysql_fetch_array($result)) {
if($row["aprobada"] == '0'){
echo '<tr class="style6">
<td  bgcolor="#FFFFFF" height="20"><span class="show">' .
$row["usuario_cedula"] .'</span>
<input type="hidden" value="' . $row["usuario_cedula"] . '" id="cedula' . $i . '"
name="cedula">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["Nombre_usuario"] .
'</span>
<input type="hidden" value="' . $row["Nombre_usuario"] . '" id="nombre' . $i . '"
name="nombre">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["Apellido_usuario"] .
'</span>
<input type="hidden" value="' . $row["Apellido_usuario"] . '" id="apellido' . $i . '"
name="apellido">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_inicial"] . '</span>
<input type="hidden" value="' . $row["hora_inicial"] . '" id="hora_inicial' . $i . '"
name="hora_inicial">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["hora_final"] . '</span>
<input type="hidden" value="' . $row["hora_final"] . '" id="hora_final' . $i . '"
name="hora_final">
</td>
<td  bgcolor="#FFFFFF"><span class="show">Pendiente</span>
<input type="hidden" value="' . $i . '" name="indentificador">
</td>
<td  bgcolor="#FFFFFF">
<input type="submit" value="Aprobar" id="' . $i . '" style=" width: 100%"
name="aprobado' . $i. '" onclick="aprobado(' . $i . ');">
<br>
<input type="submit" id="' . $i . '" name="noaprobado' . $i. '" value="Rechazar"
style=" width: 100%" onClick="desaprobar(' . $i . ');">
</td>
</tr>';
}

if($row["aprobada"] == '0'){
```

```
$i++;
}
}
@mysql_free_result($result);
?>
</tbody>
</table>
</h1>
</div>


<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SIA") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<?php
include('config.php');

$cedula            = $_POST['cedula'];
$nombre                = $_POST['nombre'];
$apellido          = $_POST['apellido'];
$logueado           = $_POST['bandera'];
$hora_inicial       = $_POST['hora_inicial'];
$hora_final         = $_POST['hora_final'];
$logueado           = $_POST['bandera'];
$logueadoDos        = 1;

if($_POST['bandera'] == '1'){

/* -- Ejecucion de la sentencia SQL - INSERT-- */

$sql    = "INSERT INTO `tiempo_permitido`(`usuario_Cedula`, `hora_inicial`,
`hora_final`, `logueado`) VALUES ('" . $cedula . "','" .$hora_inicial . "','" . $hora_final
."','" . $logueado . "')";
mysql_query($sql) or die("Error en la creacion. MySQL dice:".mysql_error());

$sql    = "UPDATE `tiempo_solicitado` SET `aprobada`='" . $logueadoDos . "'
WHERE `usuario_cedula`='" . $cedula ."' AND `hora_inicial`='" . $hora_inicial . "'
AND `hora_final`='" . $hora_final . "'";
```

```
//$sql="UPDATE `tiempo_solicitado` SET `usuario_cedula`=[value-
1],`hora_inicial`=[value-2],`hora_final`=[value-3],`aprobada`=[value-4] WHERE 1";

mysql_query($sql) or die("Error en la creacion. MySQL dice:".mysql_error());
echo "Horario asignado correctamente";


$sSQL   = "SELECT Email_usuario FROM usuario WHERE
Cedula_usuario='$cedula'";
$result = mysql_query($sSQL);
$row    = mysql_fetch_array($result);
$email  = $row["Email_usuario"];
mail($email,"Autorizacion ingreso a Observatorio UTP","Señor(a) " . $nombre . " " .
$apellido . ", Se le ha autorizado el ingreso al Domo desde " . $hora_inicial ." hasta
". $hora_final . "") ;
}else{

/* -- Ejecucion de la sentencia SQL INSERT -- */
$sql="DELETE FROM `tiempo_solicitado` WHERE `usuario_cedula`='$cedula'
AND `hora_inicial`='$hora_inicial' AND `hora_final`='$hora_final'";
mysql_query($sql) or die("Error en la creacion. MySQL dice:".mysql_error());
echo 'Horario no asignado';

$sSQL   = "SELECT Email_usuario FROM usuario WHERE
Cedula_usuario='$cedula'";
$result = mysql_query($sSQL);
$row    = mysql_fetch_array($result);
$email  = $row["Email_usuario"];
mail($email,"Autorizacion ingreso a Observatorio UTP","Señor(a) " . $nombre . " " .
$apellido . ", No ha sido autorizado el ingreso al Observatorio de la UTP, por favor
pongase en contacto con el administrador") ;
}

?>
```

Anexo 15. Código Reporte de Registros Pendientes

```
<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SIA") {
```

```
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<script>
$(document).ready(function(){
document.title = "Registros Pendientes";
});
</script>
<link href="css/contenido.css" rel="stylesheet" type="text/css" />
<script src="js/jquery-ui-1.10.3.custom/js/jquery-ui-1.10.3.custom.js"></script>
<link rel="stylesheet" href="js/jquery-ui-1.10.3.custom/css/ui-lightness/jquery-ui-
1.10.3.custom.css"></link>
<script language="javascript">

function desaprobar(posicion){
$.ajax({
data   :{  bandera            : "0",
cedula         : $('#cedula' + posicion).val(),
nombre                 : $('#nombre' + posicion).val(),
apellido       : $('#apellido' + posicion).val(),
pass           : $('#pass' + posicion).val()
},
type   :"POST",
dataType:"html",
async :false,
cache :false,
url            :"registros_pendientes_2.php",
success        :function(data){
$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
effect: "fold",
duration: 1000
}
});

againTableTwo();
},
```

```
beforeSend:function(data){$("div.utp-menu-nav-subtitulo").html("<h2>Registros
Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

function againTableTwo(){
$.ajax({
data    :{  bandera             : "0"
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"registros_pendientes.php",
success         :function(data){$("div.content").html(data);},
beforeSend:function(data){$("div.content").html("<img src=\"\">"); $("div.utp-menu-
nav-subtitulo").html("<h2>Registros Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

function aprobado(posicion){
$.ajax({
data    :{  bandera             : "1",
cedula          : $('#cedula' + posicion).val(),
nombre                  : $('#nombre' + posicion).val(),
apellido        : $('#apellido' + posicion).val(),
pass            : $('#pass' + posicion).val()
},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"registros_pendientes_2.php",
success         :function(data){
$('#dialog').html('<p>' + data + '</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
```

```
effect: "fold",
duration: 1000
}
});
againTable();
},
beforeSend:function(data){$("div.utp-menu-nav-subtitulo").html("<h2>Registros
Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

function againTable(){
$.ajax({
data    :{  bandera             : "0"},
type    :"POST",
dataType:"html",
async :false,
cache :false,
url             :"registros_pendientes.php",
success         :function(data){$("div.content").html(data);},
beforeSend:function(data){$("div.content").html("<img src=\"\">"); $("div.utp-menu-
nav-subtitulo").html("<h2>Registros Pendientes</h2>")},
error   :function(xhr, ajaxOptions, thrownerror){alert(xhr); alert(thrownerror);}
});
}

</script>
<div id="dialog"></div>
<div align="center" id="content" >
<h1 class="tabla">
<?php
include('config.php');
$user   = $_SESSION['Id'] ;
$result = mysql_query("SELECT * FROM `solicitud_registro`");
?>
<table width="600" border="1" align="center" cellpadding="0" cellspacing="1"
frame="void" bordercolor="#F3F3F3">
<thead class="style3">
<tr>
<td bgcolor="#2b57ad" align="center"><span class="show"> Cedula</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Nombre</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Apellido</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">
Institucion</span></td>
```

```
<td bgcolor="#2b57ad" align="center"><span class="show">
Comentarios</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show"> Estado</span></td>
<td bgcolor="#2b57ad" align="center"><span class="show">
Registrar</span></td>
</tr>
</thead>
<tbody>
<?php
/* -- Mostramos los registros -- */
$i  = 0;
while ($row = @mysql_fetch_array($result)) {
if($row["registrado"] == '0'){
echo '<tr class="style6">
<td  bgcolor="#FFFFFF" height="20"><span class="show">' . $row["id"] .'</span>
<input type="hidden" value="' . $row["id"] . '" name="cedula" id="cedula' . $i . '">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["nombre"] . '</span>
<input type="hidden" value="' . $row["nombre"] . '" name="nombre" id="nombre' . $i
. '">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["apellido"] . '</span>
<input type="hidden" value="' . $row["apellido"] . '" name="apellido" id="apellido' .
$i . '">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["institucion"] . '</span>
<input type="hidden" value="' . $row["institucion"] . '" name="institucion"
id="institucion' . $i . '">
</td>
<td  bgcolor="#FFFFFF"><span class="show">' . $row["comentarios"] . '</span>
<input type="hidden" value="' . $row["comentarios"] . '" name="comentarios"
id="comentarios' . $i . '">
</td>
<td  bgcolor="#FFFFFF"><span class="show">Pendiente</span>
<input type="hidden" value="' . $i . '" name="indentificador">
<input type="hidden" value="1" name="bandera"       id="bandera">
</td>
<td  bgcolor="#FFFFFF">
<input type="submit" value="Aprobar" id="' . $i . '" style=" width: 100%"
name="aprobado' . $i. '" onClick="aprobado(' . $i . ');">
<br>
<input type="submit" id="' . $i . '" name="noaprobado' . $i. '" value="Rechazar"
style=" width: 100%" onClick="desaprobar(' . $i . ');">
</td>';
$i++;
```

```
}
}
@mysql_free_result($result);
?>
</tbody>
</table>
</form>
</h1>
</div>



<?PHP
session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SIA") {
/* -- si no está logueado lo envío a la página de autentificación -- */
header("Location: index.php");
}
?>
<?php
include('config.php');

$cedula        =       $_POST['cedula'];
$logueado      =       $_POST['bandera'];
$nombre                =                $_POST['nombre'];
$apellido              =              $_POST['apellido'];
//$pass                 =                $_POST['pass'];
$logueadoDos    =       1;

if($_POST['bandera'] == '1'){

/* -- Ejecucion de la sentencia SQL__ */


$sql        = "INSERT INTO `usuario` (`Cedula_usuario`, `Nombre_usuario`,
`Apellido_usuario`, `Email_usuario`, `Password_usuario`, `Institucion`) SELECT
`id`, `nombre`, `apellido`, `email`, `Password_usuario`, `institucion` FROM
`solicitud_registro` WHERE id = '" . $cedula . "'";

mysql_query($sql ) or die("Error en la creacion. MySQL dice:".mysql_error());

$sql        = "UPDATE `solicitud_registro` SET `registrado`='" . $logueadoDos . "'
WHERE `id`='" . $cedula . "'";
```

```
mysql_query($sql ) or die("Error en la creacion. MySQL dice:".mysql_error());
echo "Usuario agregado correctamente";

$sSQL      = "SELECT Email_usuario FROM usuario WHERE Cedula_usuario='" .
$cedula . "'";
$result    = mysql_query($sSQL);
$row       = mysql_fetch_array($result);
$email     = $row["Email_usuario"];

mail($email,"Registro en el Observatorio de la UTP","Señor(a) " . $nombre . " " .
$apellido . ", Se le ha autorizado el registro como usuario del Observatorio de la
UTP. Puede ingresar con el nombre de usuario: " . $cedula . ", y su contraseña.") ;

}else{

/* -- Ejecucion de la sentencia SQL - DELETE */

$sSQL      = "SELECT email FROM solicitud_registro WHERE id='" . $cedula . "'";
$result    = mysql_query($sSQL);
$row       = mysql_fetch_array($result);
$email     = $row["email"];

$sql       =       "DELETE FROM `solicitud_registro` WHERE `id`='" . $cedula . "'";
mysql_query($sql) or die("Error en la creacion. MySQL dice:" . mysql_error());
echo "Usuario no aceptado";

mail($email,"Registro en el Observatorio de la UTP","Señor(a) " . $nombre . " " .
$apellido . ", No ha sido autorizado el registro en el Observatorio de la UTP, por
favor pongase en contacto con el administrador.") ;

}

?>
```

Anexo 16. Código Página de Control del Observatorio

```
<?php
session_start();
if ($_SESSION["autentificado"] != "SIA") {
//si no está logueado lo envío a la página de autentificación
header("Location: ../index.php");
}else{
include('../config.php');
```

```
$user                                   = $_SESSION['Id'] ;
$row['Nombre_usuario']     = $_SESSION['nom'] ;

//$result = mysql_query("SELECT COUNT( * ) c FROM tiempo_permitido WHERE
usuario_Cedula =$user AND hora_inicial <= NOW( ) AND hora_final >= NOW( )")
or die("Couldn't query the user-database.");

//$num = mysql_fetch_array($result);
$_SESSION["autentificado"] = "SIA";
/*if (!$num["c"]){
header("Location: index.php");
}*/

require("../css/cabezote.php");
}

?>
<script src="js/jquery-ui-1.10.3.custom/js/jquery-ui-1.10.3.custom.js"></script>
<link rel="stylesheet" href="js/jquery-ui-1.10.3.custom/css/ui-lightness/jquery-ui-
1.10.3.custom.css"></link>
<script src="http://200.21.217.75:5407/socket.io/socket.io.js"></script>
<script>
var objetos    = {};
var webSocket        = {};
var antDataSer       = false;
var init          = true;
$(document).ready(function(){
document.title = "Control de Observatorio";
$( "#tabs" ).tabs();
$( ".selector" ).button(); //socket.io/lib/socket.io.js


try{
webSocket = io.connect("http://200.21.217.75:5407");
console.dir(io);
}catch(err){
$('#dialog').html('<p>Servicio de puerto serial no se esta ejecutando.</p>');
$( "#dialog" ).dialog({
autoOpen: true,
modal:true,
show: {
effect: "slide",
duration: 1000
},
hide: {
```

```
effect: "fold",
duration: 1000
}
});
}

/*$('#charge').click(function(){

$('#cargars').html('<object  data="http://200.21.217.75:5406" width="1500"
height="1000"></object>');
});*/
//serial visual studio YCFHQ-9DWCY-DKV88-T2TMH-G7BHP
http://www.youtube.com/watch?v=q-llprAOO54
$('#expand').click(function(){
var caracteristicas =
        "height=1000,width=1500,scrollTo,resizable=1,scrollbars=1,location=0";
window.open("http://200.21.217.75:5406", 'Popup', caracteristicas, false);
});

$('#cesion').click(function(){
window.location.href='../cerrar_cesion.php';
});
});

function bulb(objeto){
if($("#ligthON").attr('rel') ==  'ON'){
$("#ligthON").attr('rel','OFF');
$("#ligthON").attr('src','apagado.png');
$("#ligthON").attr('id','ligthOFF');
$("#msg").text("Apagar Luz");
}else if($("#ligthOFF").attr('rel') ==  'OFF'){
$("#ligthOFF").attr('rel','ON');
$("#ligthOFF").attr('src','bulb.png');
$("#ligthOFF").attr('id','ligthON');
$("#msg").text("Encender Luz");
}
}

function gate(objeto){
if($("#openDOOR").attr('rel') ==  'OPEN'){
$("#openDOOR").attr('rel','CLOSE');
$("#openDOOR").attr('src','CerrarCompuerta.png');
$("#openDOOR").attr('id','closeDOOR');
$("#msg2").text("Cerrar compuerta");
}else if($("#closeDOOR").attr('rel') ==  'CLOSE'){
```

```
$("#closeDOOR").attr('rel','OPEN');
$("#closeDOOR").attr('src','AbrirCompuerta.png');
$("#closeDOOR").attr('id','openDOOR');
$("#msg2").text("Abrir compuerta");
}
}

function enviar(str){
/*ESPERANDO RESPUESTA DESDE EL SERVIDOR Y CUANDO RECIBA EL
EVENTO EMITIDO COMO "recibir", se ejecute la funcion recibirMensaje*/ <!--
http://200.21.217.75:5406 -->

webSocket.on("recibir",recibirMensaje);
webSocket.emit("enviar", str);
}

/*FUNCION QUE PERMITE RECIBIR MENSAJE DENTRO DE LA VARIABLE
datosServidor, Esto proviene de las decisiones del Arduino*/
function recibirMensaje(datosServidor){
window.setInterval(function(){
if(init){
$("#showResultAnt").html("<span class='ui-icon ui-icon-info' style='float: left;
margin-right: .3em;'></span><br>");
$("#showResult").html("<span class='ui-icon ui-icon-info' style='float: left; margin-
right: .3em;'>" + datosServidor.data + "</span><br>");
init = false;
}else{
$("#showResultAnt").html("<span class='ui-icon ui-icon-info' style='float: left;
margin-right: .3em;'></span>" + antDataSer + "<br>");
$("#showResult").html("<span class='ui-icon ui-icon-info' style='float: left; margin-
right: .3em;'>" + datosServidor.data + "</span><br>");
}

antDataSer = datosServidor.data;
},1000);

if(datosServidor.data == 'Join'){
window.setInterval(function(){},1000);
webSocket.emit("enviar", 'Join');
$("#domON").attr("id","domFF");
}else if(datosServidor.data == 'domFF'){
$("#domFF").attr("id","domON");
}
}
</script>
```

```
<div id="dialog"></div>
<div id="tabs">
<ul>
<li><a href="#fragment-1"><span>Video Interno</span></a></li>
<li><a href="#fragment-3" id="charge"><span>Control Telescopio</span></a></li>
<li><a href="#fragment-4"><span>Control Domo</span></a></li>
<li><a href="#fragment-5"><span>Instrucciones</span></a></li>
<li><a id="cesion"><span>Cerrar Cesión</span></a></li>
</ul>
<div id="fragment-1">
<div>
<object width="1500" height="1000"
data="http://200.21.217.75/Observatorio/node_modules/streaming/myLiveVideo.ht
m">
</object>
</div>
</div><!-- http://200.21.217.75:5406 -->
<div id="fragment-3">
<div style="text-align:center;width:942px;height:1029px;">
<a id="expand" class="selector">Expandir ventana</a>
</div>
</div>
<div id="fragment-4">
<div style="text-align:center;">
<a id="line" class="selector"><img src="bulb.png" id="ligthON"
onClick="enviar(this.id); bulb(this);" rel="ON"><br><span id="msg">Encender
luz</span></a>
<a id="line" class="selector"><img style="width:120px; height:127px;" id="domON"
src="Earth_within_celestial_sphere.gif" onClick="enviar(this.id);"><br>Mover
Domo</a>
<a id="line" class="selector"><img style="width:120px; height:127px;"
id="domOFF" src="ApagarDomo.png" onClick="enviar(this.id);"><br>Apagar
Domo</a>
<a id="line" class="selector"><img style="width:120px; height:127px;"
id="openDOOR" src="AbrirCompuerta.png" onClick="enviar(this.id); gate(this);"
rel="OPEN"><br><span id="msg2">Abrir Compuerta</span></a>
</div>
<br>
<div style="margin-left:150px; width:580px; height:200px;">
<div style="width:590px; height:155px;" class="ui-state-hover" id="results">

<div style="overflow-y:scroll; overflow-x: hidden; height:155px;"
id="showResultAnt"></div>
<br>
```

```
<div style="overflow-y:scroll; overflow-x: hidden; height:155px;"
id="showResult"></div>
</div>
</div>
</div>
<div id="fragment-5" style="text-align:center;width:942px;height:1029px;overflow-
y:scroll;overflow-x: hidden;">
PDF
</div>
</div>
<?php
require("../css/piezote.php");
?>
```

Anexo 17. Código de Flujo de Video en Streaming

```
application.onConnect = function(p_client, p_autoSenseBW)

{

//        trace("onConnect");

//Add security here
this.acceptConnection(p_client);
if (p_autoSenseBW)
this.calculateClientBw(p_client);
else
p_client.call("onBWDone");

}
Client.prototype.getStreamLength = function(p_streamName) {
trace("getStreamLength:"+p_streamName);
return Stream.length(p_streamName);

}

application.calculateClientBw = function(p_client)

{

p_client.payload = "";
for (var i=0; i<1024; i++)
p_client.payload += "1";     //2K
var res = new Object();
```

```
res.latency = 0;
res.bwTime = 0;
res.count = 0;
res.sent = 0;
res.client = p_client;
var stats = p_client.getStats();
res.beginningValues = {b_down:stats.bytes_out, b_up:stats.bytes_in, time:(new
Date()).getTime()};
res.onResult = function(p_val) {
this.count++;
var timePassed = ( (new Date()).getTime() - this.beginningValues.time );
if ( this.count == 1 )
{

this.latency = timePassed;

}

if (( this.count >= 4 && this.count < 6)&&(timePassed < 2000))

{

this.sent++;
this.client.payload += this.client.payload;
this.client.call("onBWCheck", this, this.client.payload);

}

else if ( this.sent == this.count )

{       delete this.client.payload;

var stats = this.client.getStats();
var deltaDown = (stats.bytes_out - this.beginningValues.b_down)*8/1024;
var deltaTime = (((new Date()).getTime() - this.beginningValues.time) -
(this.count - 3) * this.latency)/1000;
var kbitDown = Math.round(deltaDown/deltaTime);
trace("(d) down:"+kbitDown+" kilobits/s " );
this.client.call("onBWDone", null, kbitDown );

}

}

res.sent++;
```

```
p_client.call("onBWCheck", res, "");
for ( var k = 0; k < 3; k++ )

{

res.sent++;
p_client.call("onBWCheck", res, p_client.payload);
p_client.payload += p_client.payload;
}
}
```

Anexo 18. Código Transmisión de Video en Streaming

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Documento sin título</title>
<script src="Scripts/swfobject_modified.js" type="text/javascript"></script>
</head>

<body>

<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="342"
height="291" align="middle" id="FLVPlayer">
<param name="movie" value="FLVPlayer_Streaming.swf" />
<param name="quality" value="high" />
<param name="wmode" value="opaque" />
<param name="scale" value="noscale" />
<param name="salign" value="lt" />
<param name="FlashVars"
value="&amp;MM_ComponentVersion=1&amp;serverName=200.21.217.75&amp;
skinName=Halo_Skin_3&amp;appName=myLiveApp/instance1&amp;streamName
=mylivestream&amp;isLive=true&amp;bufferTime=0&amp;autoPlay=false&amp;aut
oRewind=true" />
<param name="swfversion" value="8,0,0,0" />

<!-- Esta etiqueta param indica a los usuarios de Flash Player 6.0 r65 o posterior
que descarguen la versión más reciente de Flash Player. -->
<param name="expressinstall" value="Scripts/expressInstall.swf" />
<!-- La siguiente etiqueta object es para navegadores distintos de IE.. -->
```

```
<!--[if !IE]>-->
<object type="application/x-shockwave-flash" data="FLVPlayer_Streaming.swf"
width="342" height="291">
<!--<![endif]-->
<param name="quality" value="high" />
<param name="wmode" value="opaque" />
<param name="scale" value="noscale" />
<param name="salign" value="lt" />
<param name="FlashVars"
value="&amp;MM_ComponentVersion=1&amp;serverName=200.21.217.75&amp;
skinName=Halo_Skin_3&amp;appName=myLiveApp/instance1&amp;streamName
=mylivestream&amp;isLive=true&amp;bufferTime=0&amp;autoPlay=false&amp;aut
oRewind=true" />
<param name="swfversion" value="8,0,0,0" />
<param name="expressinstall" value="Scripts/expressInstall.swf" />
<!-- El navegador muestra el siguiente contenido alternativo para usuarios con
Flash Player 6.0 o versiones anteriores. -->
<div>
<h4>El contenido de esta página requiere una versión más reciente de Adobe
Flash Player.</h4>
<p><a href="http://www.adobe.com/go/getflashplayer"><img
src="http://www.adobe.com/images/shared/download_buttons/get_flash_player.gif"
alt="Obtener Adobe Flash Player" /></a></p>
</div>
<!--[if !IE]>-->
</object>
<!--<![endif]-->
</object>
<script type="text/javascript">
swfobject.registerObject("FLVPlayer");
</script>
</body>
</html>
```

Anexo 19. Código Control VNC

```
Public Class Form1

' Declares ThinVnc and ScreenScraper components
Public WithEvents tvnc As ThinVncX.ThinVnc = New ThinVncX.ThinVnc()
Dim WithEvents ws As ThinVncX.ThinWebServer

Dim pid, pidTwo, pidThree, pidFour As Integer
```

```
Dim bandera As String = ""

Dim scraper As ThinVncX.ThinScreenScraper

Public Sub New()
' This call is required by the Windows Form Designer.
InitializeComponent()

' Instanciates a new ThinVnc and ScreenScraper component

scraper = New ThinVncX.ThinScreenScraper()

' Tells ThinVnc to use ScreenScraper
tvnc.Scraper = scraper

'Capturando Puerto
If Not tvnc.WebServer.HttpActive Then
tvnc.WebServer.HttpPort = CInt(textPort.Text)
'MessageBox.Show(textPort.Text)
End If

Try
'Capturando Puerto
tvnc.WebServer.HttpActive = Not tvnc.WebServer.HttpActive
If tvnc.WebServer.HttpActive Then
bandera = "INICIO"
'bStart.Text = "Stop"
Else
bandera = "NOFUNCT"
'bStart.Text = "Start"
End If

Catch Ex As System.Exception
MessageBox.Show(Ex.Message)
End Try

End Sub

Function Listar(ByVal validation As String) As Integer

Dim p As Process
Dim text As Integer = 0
Dim gla As Integer = 1
Dim indicador As Boolean = True
```

```
If "AutostarSuite" <> validation Then
While gla = 1
For Each p In Process.GetProcesses()
If Not p Is Nothing Then
'ListBox1.Items.Add(p.ProcessName)
If p.ProcessName = validation Then
ListBox1.Items.Add(" Encontro PID: " & p.Id)
text = p.Id
scraper.AddProcessId(text)
ListBox1.Items.Add(" Agrego el proceso")
gla = 2
Else
indicador = False
End If
End If
'ListBox1.Items.Add("Buscando")
Next

End While
Else
While gla < 50
For Each p In Process.GetProcesses()
If Not p Is Nothing Then
ListBox1.Items.Add(p.ProcessName)
If p.ProcessName = validation Then
ListBox1.Items.Add(" Encontro PID: " & p.Id)
scraper.AddProcessId(p.Id)
ListBox1.Items.Add(" Agrego el proceso")
Else
indicador = False
End If
End If
Next
gla = gla + 1
End While
End If
ListBox1.Items.Add("Saliendo.....")
Return indicador

End Function

Function ListarAutoStar(ByVal validation As String) As Integer

Dim Lista_Procesos() As Process
```

```
Lista_Procesos = Process.GetProcessesByName("AutostarSuite")

For Each p In Lista_Procesos
ListBox1.Items.Add(p.ProcessName)
ListBox1.Items.Add(" Encontro PID: " & p.Id)
scraper.AddProcessId(p.Id)
ListBox1.Items.Add(" Agrego el proceso")
Next
End Function

Private Sub CaptureModeChanged(ByVal validation As String)

Dim muestraCcops As Boolean = True
Dim muestraAutoStar As Boolean = True

If IsNothing(scraper) Then Exit Sub

If validation = "INICIO" Then ' Evento de recargar

'System.Diagnostics.Process.Start(Environ("windir") & "\system32\cmd.exe",
"taskkill /im virtualdj_home.exe")
'Shell("cmd /c taskkill /im wordpad.exe", vbHide)
'Shell("cmd /c taskkill /im EasyWeather.exe", vbHide)
'Shell("cmd /c taskkill /im AutostarSuite.exe", vbHide)
'Environ("windir") & "\system32\calc.exe C:\Program Files
(x86)\VirtualDJ\virtualdj_home.exe
'System.Diagnostics.Process.Start(Environ("windir") & "\system32\cmd.exe",
"taskkill /im calc.exe")
ListBox1.Items.Add("Inicio servicio remotamente.")

'muestraCcops = Listar("Ccdops")
'If muestraCcops = True Then
'ListBox1.Items.Add("Encontro con Ccdops")
'Else
pidFour = CType(Shell("C:\prueba.bat", vbHide), Integer)
ListBox1.Items.Add("Ejecuto Ccdops")
muestraCcops = Listar("Ccdops")
ListBox1.Items.Add("Termino con ccdops")
'End If

'pidTwo = Listar("AutostarSuite")
'If muestraAutoStar = True Then
'ListBox1.Items.Add("Encontro Autostar")
'Else
pidTwo = CType(Shell("C:\prueba2.bat", vbHide), Integer)
```

```
ListBox1.Items.Add("Ejecuto AutostarSuite")
pidTwo = Listar("AutostarSuite")
ListBox1.Items.Add("Termino con autostar")
pidThree = CType(Shell("C:\Program Files\EasyWeather\EasyWeather.exe",
vbMaximizedFocus), Integer)
scraper.AddProcessId(pidThree)
'pidTwo = Listar("explorer")
'End If

scraper.ExcludeProcesses = False
End If
End Sub

Private Sub tvnc_OnSessionStarted() Handles tvnc.OnSessionStarted
ListBox1.Items.Add("INICIO ... " & bandera)
If bandera <> "" Then
CaptureModeChanged(bandera)
End If
End Sub

End Class
```

Anexo 20. Código Cerrar Sesión.

```
<?PHP
include('config.php');

session_start();
date_default_timezone_set('America/Bogota');
if ($_SESSION["autentificado"] != "SI") {
/* -- Si no está logueado se envia a la página de autentificación --*/
header("Location: index.php");
}

$hora_salida =      date("Y-m-d H:i:s");
$user               =       $_SESSION['Id'];
$hora_entrada       =       $_SESSION["ultimoAcceso"];
$sSQL                       =       "UPDATE log SET hora_salida='" . $hora_salida .
"' WHERE id_usuario=" . $user ." AND hora_entrada='" . $hora_entrada . "'";

mysql_query($sSQL) or die("Error actualizando log. MySQL dice:" .$sSQL.
mysql_error());
$_SESSION           =       array();
```

```
session_destroy();

/* -- Se redirecciona a la página inicial index.php  -- */
echo '<script>
window.location="index.php"
</script>';
?>
```